IMPROVING THE COMPUTATIONAL EFFICIENCY OF APPROXIMATE DYNAMIC PROGRAMMING USING NEURAL NETWORKS, WITH APPLICATION TO A MULTI-RESERVOIR HYDROPOWER AND WIND POWER SYSTEM

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Kyle Robert Perline

December 2017

IMPROVING THE COMPUTATIONAL EFFICIENCY OF APPROXIMATE DYNAMIC PROGRAMMING USING NEURAL NETWORKS, WITH APPLICATION TO A MULTI-RESERVOIR HYDROPOWER AND WIND POWER SYSTEM

Kyle Robert Perline, Ph.D.

Cornell University 2017

This dissertation develops a more computationally efficient Approximate Dynamic Programming (ADP) method that can be applied to optimal control problems that are stochastic, nonconvex, and either finite or rolling horizon with many stages. The primary application is to a realistic model of Bonneville Power Administration, which is a large wind power and hydropower producer in the Pacific Northwest. The objective is to determine the optimal amount of power to buy and sell on the day and hour ahead markets conditioned on the wind power forecast, as well as how much water to release from each reservoir.

The second Chapter in this dissertation develops the Fitting via Unimodal Approximation Optimization (FUA) method for more accurately approximating the value function in ADP with a Feedforward Neural Network with one hidden layer (FFNN1). A major part of FUA is the new Unimodal Approximation Optimization (UAO) algorithm that is used to perform FFNN1 hyperparameter optimization. UAO can be applied to optimization problems with a discrete domain and a noisy unimodal objective function, and it is proven that UAO converges almost surely to the correct solution. Results on two control problems with 4, 12 and 15 state space dimensions show that approximating the value function in ADP with an FFNN1 using FUA yields a more accurate control solution in less time as compared to

using other methods of fitting an FFNN1.

Chapter 3 presents the Long Term Generation method that generates long-term synthetic wind power scenarios conditioned on historical sequential short-term wind power forecasts. Power systems with wind power integration can be simulated on this data in order to more accurately evaluate the performance of their control policies. Additionally, the Joint Distribution Comparison test is developed to evaluate the quality of these synthetic scenarios.

Finally, in Chapter 4 a stochastic rolling horizon model of Bonneville Power Administration is developed and the resulting control problem is solved using the ADP algorithm developed in Chapter 2 and is evaluated using data generated in Chapter 3. The stochastic control formulation has a nonlinear objective function, 24 decision variables, and 16 state space dimensions.

BIOGRAPHICAL SKETCH

Kyle Perline was born in Burlington, Vermont on October 24, 1989 to Kevin Perline and Lori Rippa. Kyle has always had a love of mathematics, engineering, and physics, which led him to attend the Honors Program at Clarkson University from 2008 to 2012. There, he obtained a double Bachelor of Science Degree in Physics and Mathematics, as well as a double Bachelor of Science Degree in Aeronautical Engineering and Mechanical Engineering. In the summer of 2012 Kyle first went to Cornell and began work with his adviser Christine Shoemaker.

This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

Thank you to my adviser Christine Shoemaker. Throughout five and a half years she helped me focus my research and yet gave me enough flexibility to explore my changing interests.

I am grateful to the entire Center for Applied Mathematics. This includes all of the staff, directors, and in particular my committee members, Alexander Vladimirsky and Huseyin Topaloglu, who help make CAM an excellent academic program. Of course, my time there was so enjoyable largely because of my fellow CAM students, and I thank all of them for giving me motivation to go to the office.

I would also like to thank everyone I worked with during my internship at The Aerospace Corporation during the Spring of 2016.

Finally, I want to share my appreciation for my entire family.

This work was supported by the National Science Foundation Grant DGE-1650441, the Center for Applied Mathematics, and teaching assistantships in Cornell's Mathematics Department.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The optimal control problems of power systems are often formulated so that they are either linear or convex. However, these formulations usually need to simplify the power system control problem by using linear or convex approximations. For example, hydropower production is a nonconcave function of water head and flow rate, but it is common to use a linear, piecewise linear, or concave approximation of these power curves when planning hydropower operations. In this dissertation we develop a more efficient approximate dynamic programming algorithm that can be used to solve stochastic nonconvex optimal control problems in general, and, in particular, coordinate hydropower production to help integrate wind energy into the power grid.

Wind power capacity has been substantially increasing both in the United States and abroad [1]. While it is desirable to continue to increase wind power capacity in order to provide more renewable energy sources, it is difficult to integrate large amounts of wind power into the power grid. The two primary difficulties are that wind power is non-dispatchable, meaning it is not controllable, and that it cannot be accurately forecasted. This results in situations where the wind power may suddenly and unexpectedly increase or decrease, and the power grid must be able to accommodate these unpredictable ramps.

A common approach to integrating wind power is to combine, or balance, it with another dispatchable power source [14, 15]. The combined power generation can be more easily and reliably integrated into the power grid. Hydropower in particular is a common power source to balance wind power because it has a large storage capacity and a quick response time [5, 2].

Most wind power research focuses on power producers that are price takers, meaning the amount of power that they sell on the power market does not impact

the price of power. This is true for most producers, which produce a relatively small amount of power as compared to the power markets. However, some producers are large enough where this assumption is not valid.

The primary example investigated in this thesis is based on the Bonneville Power Administration (BPA). BPA is a federal agency in the US Pacific Northwest that operates hydropower reservoirs along the Columbia and Snake Rivers. Additionally, BPA markets this produced hydropower, which has a capacity of almost 20,000 MW, as well as the wind power produced in the geographic area, which has a capacity of about 4700 MW.

BPA markets a large enough amount of power that it is modeled as a price setter. We propose an extension of the model of BPA developed in [12]. In this rolling horizon model, BPA participates in an hourly day ahead market and an hourly intraday market, and BPA's actions influence the price of power in both markets. The resulting optimal control model is stochastic because the controls are conditioned on an uncertain wind forecast, and the model is nonconvex because the hydropower generation curves are nonconcave.

The primary differences in the dynamic programming formulation and power producer model presented here as compared to [12] are: the hydropower system includes seven reservoirs instead of two; the hydropower generation curves are nonconcave; the wind power forecasts use historical forecasts generated by BPA; the rolling horizon formulation enacts decisions every eight hours instead of every 24; and the dynamic programming formulation includes in the state space the day ahead power committed during previous days. Additionally, two models are compared. In the Combined model the wind and hydropower are marketed by the same power producer, while in the Individual model a separate wind power

producer and a separate hydropower producer act independently. In both models all power producers have market influence.

In Chapter 2 we first develop a more efficient approximate dynamic programming (ADP) algorithm to solve this optimal control problem. The ADP algorithm can be used to solve stochastic and nonconvex optimal control problems, but it has an exponential computational cost with respect to the number of state space dimensions. Much work has been performed to create a more accurate value function approximation by using various methods of sampling the state space and different classes of functions for approximating the value function. For example, some of the classes of functions used to approximate the value function include cubic piecewise and tensor polynomials [10], Multiadaptive Regression Splines (MARS) [8], and a local approximator called a Nadaraya Watson model [6].

We develop a more accurate method of fitting a Feedforward Neural Network with one hidden layer (FFNN1) to the value function in order to increase the ADP algorithm efficiency. Neural networks have been used as the value function approximation [4, 7, 9]. However, there are many ways to fit an FFNN1 to a dataset, which in the case of ADP consists of the state space/value function samples. There has been relatively little work investigating the impact on ADP solutions when different fitting methods are used. In particular, FFNN1s have a single hyperparameter called the number of hidden nodes that should be tuned, and fitting an FFNN1 is a noisy process, meaning fitting two FFNN1s with the same hyperparameter to the same data will likely not yield the same trained FFNN1s.

We develop the Fitting via Unimodal Approximation Optimization (FUA) method of fitting an FFNN1 that is specifically tailored for use in ADP. The primary component of FUA is to use the new Unimodal Approximation Optimization

(UAO) algorithm to perform a hyperparameter optimization. Additionally, we also demonstrate that, contrary to common practice, it is better to only partition the training data into a training and validation set, and to not use a test set.

In Chapter 3 we develop the Long Term Generation method of generating wind scenarios to help evaluate the ADP solutions obtained on the wind and hydropower problem. Power systems with wind integration are often evaluated using simulation. In this rolling horizon process, the power system controls are calculated at each time step conditioned on the most recent wind power forecast. The state of the system and the cost incurred then evolve according to the actual wind power outcome. This means that a sequence of wind power forecasts are required to calculate the controls and a corresponding sequence of wind power outcomes over the same horizon is required to evaluate the system.

There are three common methods for obtaining these wind power forecast and outcome sequences. First, historical data can be used, which gives a single sequence of data [3]. Second, a sequence of wind power outcomes is obtained, either by gathering historical data or by generating a sequence from a stochastic process. Synthetic wind forecasts are then generated conditioned on the outcome sequence [13]. However, this approach assumes the forecast errors of forecasts generated at different time steps are independent, which may not be correct. And third, a single short-term historical wind power forecast is obtained and then short-term wind outcome scenarios are generated conditioned on this forecast [11].

The Long Term Generation method generates long-term wind scenarios conditioned on a sequence of short-term wind power forecasts. There is an advantage of using LTG over each of the three current approaches mentioned above. First, LTG can augment historical data used in the first approach in order to more ac-

curately estimate the distribution of performances. Second, LTG does not require estimating a multivariate time series, like in the second approach. And third, LTG generates arbitrarily long-term scenarios, and so these scenarios can be used to help evaluate the long-term performance of a power system; the third approach mentioned above only generates scenarios over the same horizon as the forecast.

The Joint Distribution Comparison (JDC) test is also developed here to help evaluate these scenarios. Existing scenario evaluation methods only compare the synthetic scenarios to historical wind outcomes. The JDC test instead compares the joint distribution of historical forecasts and historical outcomes to the joint distribution of historical forecasts and synthetic scenarios.

Finally, Chapter 4 presents the model of a wind and hydropower producer with market influence. The FUA algorithm developed in Chapter 2 is used to solve the resulting optimal control problem. The power system is evaluated using wind scenarios generated by the LTG method developed in Chapter 3. A case study compares the Combined model, in which a single company markets both the wind and hydropower, and the Individual model, in which an individual wind power company and hydropower company act independently. Empirical results show that the differences between these two models are magnified when the power producers have greater market influence.

BIBLIOGRAPHY

[1] Global Wind Energy Council, Global Statistics. `http://www.gwec.net/global-figures/graphs/`. Accessed: 2017-08-16.

[2] Jorge Márquez Angarita and Julio Garcia Usaola. Combining hydrogeneration and wind energy: Biddings and operation on electricity spot markets. *Electric Power Systems Research*, 77(5):393–400, 2007.

[3] Rüdiger Barth, Heike Brand, Peter Meibom, and Christoph Weber. A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on*, pages 1–8. IEEE, 2006.

[4] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

[5] Edgardo D Castronuovo and JA Peças Lopes. On the optimization of the daily operation of a wind-hydro power plant. *IEEE Transactions on Power Systems*, 19(3):1599–1606, 2004.

[6] Cristiano Cervellera, Mauro Gaggero, and Danilo Macciò. Low-discrepancy sampling for approximate dynamic programming with local approximators. *Computers & Operations Research*, 43:108–115, 2014.

[7] Cristiano Cervellera, Aihong Wen, and Victoria CP Chen. Neural network and regression spline value function approximations for stochastic dynamic programming. *Computers & operations research*, 34(1):70–90, 2007.

[8] Victoria CP Chen, David Ruppert, and Christine A Shoemaker. Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming. *Operations Research*, 47(1):38–53, 1999.

[9] Huiyuan Fan, Prashant K Tarun, and Victoria CP Chen. Adaptive value function approximation for continuous-state stochastic dynamic programming. *Computers & Operations Research*, 40(4):1076–1084, 2013.

[10] Sharon A Johnson, Jery R Stedinger, Christine A Shoemaker, Ying Li, and Jose Alberto Tejada-Guibert. Numerical solution of continuous-state dy-

namic programs using linear and spline interpolation. *Operations Research*, 41(3):484–500, 1993.

[11] Pierre Pinson, Henrik Madsen, Henrik Aa Nielsen, George Papaefthymiou, and Bernd Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind energy*, 12(1):51–62, 2009.

[12] Sue Nee Tan. *Computationally Efficient Hydropower Operations Optimization for Large Scale Cascaded Hydropower Systems Reflecting Market Power, Fish Constraints, Multi-Turbine Powerhouses, and Renewable Resource Integration*. PhD dissertation, Cornell University, 2017.

[13] Aidan Tuohy, Peter Meibom, Eleanor Denny, and Mark O'Malley. Unit commitment for systems with significant wind penetration. *IEEE Transactions on Power Systems*, 24(2):592–601, 2009.

[14] Bart C Ummels, Madeleine Gibescu, Engbert Pelgrum, Wil L Kling, and Arno J Brand. Impacts of wind power on thermal generation unit commitment and dispatch. *IEEE Transactions on energy conversion*, 22(1):44–51, 2007.

[15] Chaoyue Zhao, Qianfan Wang, Jianhui Wang, and Yongpei Guan. Expected value and chance constrained stochastic unit commitment ensuring wind power utilization. *IEEE Transactions on Power Systems*, 29(6):2696–2705, 2014.

CHAPTER 2

IMPROVED STOCHASTIC NEURO-DYNAMIC PROGRAMMING WITH APPLICATIONS TO HYDROPOWER

2.1 Abstract

Hydropower operation and other control problems can be analyzed using finite horizon Approximate Dynamic Programming (ADP), but more computationally efficient ADP algorithms are required for systems with many dimensions, e.g. reservoirs. In Neuro-Dynamic Programming, neural networks (commonly Feedforward Neural Networks with one hidden layer (FFNN1)) are used to approximate the value function. However, the FFNN1 accuracy, and therefore the control solution accuracy, is stochastic and depends upon the FFNN1's hyperparameter. We increase ADP solution accuracy without increasing computation time by developing the Fitting via Unimodal Approximation Optimization (FUA) algorithm to create more accurate FFNN1 value function approximations. FUA uses the new Unimodal Approximation Optimization (UAO) algorithm to perform FFNN1 hyperparameter optimization. It is proven that UAO converges almost surely, and UAO outperforms other algorithms, including Bayesian Optimization, on the tested FFNN1 hyperparameter optimization problems. We develop the CAA and SAA methods of assessing the quality of an ADP control solution. The CAA and SAA results on two hydropower control problems and one inventory control problem show that fitting the value function with an FFNN1 using FUA yields statistically more accurate control solutions using less computation time when compared to other methods of fitting an FFNN1.

2.2 Introduction

Hydropower is an important renewable energy source that can reduce the need for fossil fuels. The operation of hydropower facilities is often formulated as the so-

lution to a discrete-time, stochastic, finite horizon, and nonlinear optimal control problem that can be difficult to solve. Stochastic Dynamic Programming (SDP) can be used to solve these types of problems, and it has been a focus in hydropower analysis for decades [37, 38]. However, SDP suffers from the 'curse of dimensionality,' meaning its computational cost is exponential with respect to the number of state dimensions.

Approximate Dynamic Programming (ADP) reduces the computational cost by generating an approximation of the future value function and efficiently sampling the state space [30]. The goal of solving larger, more realistic hydropower problems has therefore provided motivation for developing ADP algorithms [22]. However, even ADP is challenged by high dimensional problems.

Various models have been used to approximate the value function when ADP is applied to hydropower problems. These include Hermite polynomials [19]; cubic piecewise and tensor polynomials [22]; Multivariate Adaptive Regression Splines (MARS) [10], which was also applied in inventory problems [13, 11, 12]; local approximators called a Nadarya-Watson model [8]; and Feedforward Neural Networks (FFNNs), in which case the ADP algorithm is sometimes called Neuro-Dynamic Programming [4, 6].

More commonly, FFNNs with one hidden layer (FFNN1) are used, which have a single integer-valued hyperparameter called the number of hidden nodes. These were applied in both hydropower problems [10] and inventory problems [16].

Using FFNN1s, as well as FFNNs more generally, as the future value function approximation can yield accurate control solutions, but there are two main problems that make it difficult to reliably use them in ADP applications. First,

the selection of the FFNN1 hyperparameter can have a significant impact on how well a trained FFNN1 approximates a value function. And second, training an FFNN1 is stochastic, meaning that it is not necessarily the case that training two FFNN1s with the same hyperparmeter on the same data will yield the same trained FFNN1s. This means that the ADP solution accuracy is stochastic and depends on the hyperparameter selection.

In the neural network community, hyperparemter optimization methods have been developed to more reliably train an accurate FFNN. In a process we refer to as 'fitting,' the hyperparamters are first optimized as multiple FFNNs are sequentially trained, and then upon termination the trained FFNNs with the smallest estimated errors are selected [5].

This process can be decomposed into four Components: Component C1 is the method of training a single FFNN with fixed hyperparameters on a dataset; Component C2 is the method of estimating the error of a trained FFNN; Component C3 is the hyperparameter optimization algorithm used to determine the optimal hyperparameters; and Component C4 is the number of trained FFNNs that are combined together in an ensemble to create the value function approximation. There are multiple methods that can be selected for each Component, so a fitting algorithm consists of a selection of one method per Component.

Our objective is to create an FFNN1 fitting algorithm that increases the accuracy of ADP solutions, when applied to hydropower problems and general control problems, using less computation. The first step is to develop the new Unimodal Approximation Optimization (UAO) algorithm to apply in the hyperparameter optimization, Component C3, and then to determine the best combination of methods for all four Components. To evaluate results we develop the new Statistical Accu-

racy Analysis (SAA) and Computation and Accuracy Analysis (CAA) tests. This research focuses on FFNN1s, although it can be extended to more general FFNNs.

First main contribution: We develop the UAO algorithm and apply it to hyperparameter optimization. The domain of this optimization problem is the positive integers (i.e. number of hidden nodes), and the noisy objective function is the expected estimated error. The two optimization algorithms that have already been developed specifically for tuning the number of hidden nodes in an FFNN1 are a two-phase greedy process [33] and a manual optimization based on a coarse-to-fine approach [36, 15]. However, most work in the neural network community focuses on performing multivariate hyperparameter optimization on general FFNNs when they are trained with millions of data points. Research has shown that Bayesian Optimization outperforms random sampling, which in turn outperforms grid sampling [2, 3, 34].

We developed the UAO algorithm because of the empirical observation that a unimodal function, with respect to the number of hidden nodes, could alway be fit within the 99% confidence interval of the expected error when training an FFNN1 on our test problems. Existing optimization algorithms suitable for a discrete domain and noisy unimodal objective function were designed for the bandit problem. However, these should not be applied in FFNN1 hyperparameter optimization because they either assume the noise follows a Bernoulli distribution or they require too many evaluations (number of trained FFNN1s) to be applicable [20, 14, 21].

The UAO algorithm is designed to require few evaluations by using a unimodal surrogate, or meta-model, to approximate the objective function. This surrogate then helps determine the number of hidden nodes with which to train the next FFNN1. Our empirical testing shows that UAO outperforms other algorithms,

including Bayesian Optimization, on FFNN1 hyperparameter optimization. We prove that UAO converges almost surely when the noisy objective function is unimodal and the noise distributions have compact support. This result is also extended to the multidimensional domain.

Second main contribution: We determine the best method for each of the four fitting Components, C1 through C4. This results in the Fitting via Unimodal Approximation Optimization (FUA) algorithm. The specific Components of FUA are: for Component C1 use the Early Stopping method (and not Bayesian Regularization) to train individual FFNN1s; for Component C2 estimate the error using the validation set (and not a test set); for Component C3 use the UAO algorithm (and not Bayesian Optimization, greedy search, or random search); and for Component C4 use the single best-trained FFNN1 (and do not use an ensemble).

Third main contribution: We perform testing on two hydropower control problems and an inventory control problem, and we develop the Statistical Accuracy Analysis (SAA) and Computation and Accuracy Analysis (CAA) methods to evaluate the accuracy and computational cost of an ADP control solution. Both SAA and CAA are related to, but more statistically extensive than, the evaluation process used in [16]. In SAA, the ADP control policy is determined over multiple trials. For each trial, its performance is calculated by simulating the system over many trials with appropriate stochastic inputs over all time steps. SAA tests the null hypothesis that two methods of determining the ADP control policy yield the same mean performance. The CAA method plots the trade-off between computation time and accuracy and shows the distribution over these two results. Results on test problems show that approximating the ADP value function with an FFNN1 using FUA yields a more accurate control solution in less time than alternative

methods of fitting an FFNN1.

This Chapter is organized as follows. First, a brief background on ADP and a detailed explanation of the four Components comprising a fitting algorithm are described in Section 2.3. In Section 2.4, the Unimodal Approximation Optimization algorithm is first motivated and developed. Then, to help reduce the large number of FFNN1 fitting algorithms that need to be compared based on their impact on ADP solutions, a subset of fitting algorithms are first compared based on how well they fit FFNN1s to value functions. In Section 2.5 two hydropower control problems, with 4 and 12 state dimensions, and one inventory control problem are presented. In Section 5 four fitting algorithms are used to approximate the value function in the control problems with an FFNN1, and they are compared using the SAA and CAA fitting assessments. Conclusions are summarized at the end.

2.3 Background on Feed Forward Neural Networks and Research Plan

It is helpful to first provide an overview of finite horizon Approximate Dynamic Programming in order to make it clear how FFNN1s are used. In a T-stage discrete-time control problem, at stage k a real-valued cost $c^k(s^k, \pi^k(s^k), \omega^k)$ is incurred for being in the N-dimensional state s^k belonging to the state space $\mathcal{S}^k \subset \mathbb{R}^N$, taking action $\pi^k(s^k)$ in a set of allowable actions $\Pi^k(s^k)$, and obtaining sample ω^k from an exogenous random variable. At the terminal stage the cost is $c^T(s^T)$. The state evolves according to the transition function $s^{k+1} = g^k(s^k, \pi^k(s^k), \omega^k)$. In backwards dynamic programming the optimal actions are determined by setting the terminal value function $J^T(s^T) = c^T(s^T)$ and for $k = T-1, ..., 1$ recursively calculating the

value function J^k at stage k as

$$J^k(s^k) = \min_{\pi^k(s^k)\in\Pi^k(s^k)} \Big\{ \mathop{\mathbb{E}}_{\omega^k} \left[c^k(s^k, \pi^k(s^k), \omega^k) + J^{k+1}(s^{k+1}) \right] : \\ s^{k+1} = g^k(s^k, \pi^k(s^k), \omega^k) \Big\}. \tag{2.1}$$

The optimal action is then the argument solving Equation (2.1). In ADP the value function $J^k : \mathcal{S}^k \to \mathbb{R}$ is approximated by numerically evaluating $J^k(s_i^k)$ for some set $\{s_i^k\}_{i=1}^L \subset \mathcal{S}^k$ and fitting a function approximation $\hat{J}^k : \mathcal{S}^k \to \mathbb{R}$ to the resulting dataset $\mathcal{D}^k = \{(s_i^k, J^k(s_i^k))\}_{i=1}^L$.

Feedforward Neural Networks with one hidden layer $F(\cdot;\theta,h) : \mathcal{S} \to \mathbb{R}$ can be fit to the datasets $\mathcal{D}^k$ to approximate the value functions. FFNN1s have a single positive integer-valued hyperparameter h called the number of hidden nodes and are parameterized by a vector θ. The number of hidden nodes h controls the dimension of θ, and so h controls the complexity of an FFNN1. For a given h and θ the mean squared L_2 generalization error of an FFNN1 $F^k(\cdot;\theta,h)$ is

$$GE(F^k, J^k) = \frac{1}{\int_{s^k\in\mathcal{S}^k} ds^k} \int_{s^k\in\mathcal{S}^k} (J^k(s^k) - F(s^k;\theta,h))^2 ds^k. \tag{2.2}$$

Since J^k is not known over all $\mathcal{S}^k$ this cannot be computed, so the error can be estimated as

$$\widehat{GE}(F^k, \mathcal{D}^k) = \frac{1}{L} \sum_{s^k\in\mathcal{D}} (J^k(s^k) - F(s^k;\theta,h))^2. \tag{2.3}$$

An FFNN1 F^k with a fixed h is trained on a dataset $\mathcal{D}^k$ by solving the minimization problem

$$\min_\theta \widehat{GE}(F^k(\cdot;\theta,h), \mathcal{D}^k). \tag{2.4}$$

Fitting an FFNN1 to a dataset $\mathcal{D}$ is a two-step process that requires four components: 1) Select a method of training a single FFNN1 (Component 1) on $\mathcal{D}$

and use a hyperparameter optimization algorithm (Component 3) to train multiple FFNN1s while optimizing the hyperparameters; and 2) Use an error estimation method (Component 2) to determine which FFNN1s are the most accurate and combine the best-trained FFNN1s into an ensemble (Component 4).

2.3.1 Component C1: Training Algorithms ES and BR

Solving the optimization problem in Equation (2.4) often results in overfitting due to the large number of parameters in θ. In the **Early Stopping** (ES) method $\mathcal{D}$ is partitioned into a training set $\mathcal{T}$ and validation set $\mathcal{V}$; a local optimization algorithm solves Equation (2.4) to minimize the training error $\widehat{GE}(F^k(\cdot;\theta,h),\mathcal{T})$; and the optimization is terminated when the validation error $\widehat{GE}(F^k(\cdot;\theta,h),\mathcal{V})$ begins to increase, which indicates the FFNN1 is overfitting $\mathcal{T}$ [5]. The **Bayesian Regularization** (BR) method penalizes non-zero components of θ, which drives some parameters to zero and thereby reduces the complexity and helps to prevent overfitting [26]. BR does not require a separate validation set and should be solved to convergence.

An important practical consideration is that training a single FFNN1 with BR requires significantly more computation time than ES. Results in Section 2.6 show that the computation time can be over 100 times greater. Because of the computation time considerations in ADP the fitting process of training multiple FFNN1s will only be applied when the ES method is used.

Training is a noisy process. If FFNN1s F_i, $i = 1, 2, ...$ with h (fixed) hidden nodes are trained on the same data $\mathcal{D}$ and θ_i are the trained FFNN1 parameters of F_i, then it is not necessarily the case that $\theta_i = \theta_j$. The first reason is that in ES

$\mathcal{D}$ is randomly partitioned into new sets $\mathcal{T}$ and $\mathcal{V}$ when training a new FFNN1. Second, the objective function in Equation (2.3) has multiple local minima, so if the parameters θ are randomly initialized the local optimization algorithm will converge to a random local minimum [32, 17, 24]. So even for a fixed h and partition $\mathcal{T}$ and $\mathcal{V}$ the trained parameters θ_i can be different.

2.3.2 Component C2: Generalization Error Estimation methods TSM and VSM

Two estimation methods are investigated when using ES training. The **Test Set Method** (TSM) first partitions $\mathcal{D}$ into a training set $\mathcal{T}$, validation set $\mathcal{V}$, and test set $\mathcal{E}$ in a ratio of 70\15\15, the ES training is performed as described using $\mathcal{T}$ and $\mathcal{V}$, and the generalization error is estimated with the test set error $\widehat{GE}(F, \mathcal{E})$. In the **Validation Set Method** (VSM) $\mathcal{D}$ is only partitioned into $\mathcal{T}$ and $\mathcal{V}$ in a ratio of 80\20 and the validation error $\widehat{GE}(F, \mathcal{V})$ is the estimate. The trade-off is that the Test Set Method uses fewer samples in $\mathcal{T}$ and $\mathcal{V}$, possibly making it less likely to train an FFNN1 with a small L_2 error, but the independent test set error is likely more accurate. When using BR training only one FFNN1 is trained so there is no need to estimate the error.

2.3.3 Component C3: Hyperparameter Optimization algorithms UAO, BO, G, R

Suppose a dataset $\mathcal{D} = \{(s_i, J(s_i)) : s_i \in \mathcal{S}\}_{i=1}^{L}$ is drawn from a value function $J : \mathcal{S} \to \mathbb{R}$ and an FFNN1 F with h hidden nodes is trained using ES and error

estimation method C2. The estimated error ($\widehat{GE}(F, \mathcal{E})$ if C2=TSM or $\widehat{GE}(F, \mathcal{V})$ if C2=VSM) is a random variable

$$X_h^{\mathcal{D},C2} \tag{2.5}$$

with a distribution that depends on h. For a set of candidate number of hidden nodes $\mathcal{H}$ the hyperparmeter optimization can be formulated as minimizing the expected estimated error

$$\min_{h \in \mathcal{H}} \mathbb{E}(X_h^{\mathcal{D},C2}). \tag{2.6}$$

This hyperparameter optimization problem has a discrete domain (number of hidden nodes), is noisy (training an FFNN1 yields a sample from $X_h^{\mathcal{D},C2}$), and is computationally expensive. The hyperparameter optimization algorithms that are tested are **Unimodal Approximation Optimization** (UAO), **Bayesian Optimization** (BO), **Greedy** (G), and **Random** (R).

2.3.4 Component C4: Ensemble

Theory proves that an average of multiple FFNN1s can have a smaller generalization error than any of the FFNN1s individually, and ensembles are often used in practice [25]. The trade-off in ADP is that an ensemble can yield a more accurate value function approximation, but it will take more computation time to evaluate this approximation in Equation (2.1).

2.3.5 Summary: $Fit(C1, C2, C3, C4)$

A fitting algorithm can therefore be written as $Fit(C1, C2, C3, C4)$ for each choice of Components C1 through C4. As previously mentioned the Bayesian Regulariza-

tion (BR) training method requires significant computation time, and so the fitting process will consist of training exactly one FFNN1 when C3=BR. When using ES to train the FFNN1s there are four optimization algorithms (UAO, BO, G, R), two error error estimation methods (TSM, VSM), and the ensemble can consist of $1, 2, ...$ trained FFNN1s. Testing in Section 2.4.3 will show that the **Fitting via Unimodal Approximation Optimization** method $Fit(ES, VSM, UAO, 1)$, or FUA, that uses Early Stopping for C1, the Validation Set Method for C2, the UAO hyperparamter optimization algorithm for C3, and an ensemble of one for C4 outperforms the rest.

2.3.6 Implementation

The FFNN1s are all implemented in MATLAB with the `feedforwardnet` function and use the default settings unless otherwise noted. In particular, the training algorithm is switched between using the Levenberg-Marquardt training algorithm when Early Stopping is used (there is no need for stochastic gradient descent because the number of data points, i.e. state space samples, is relatively small) and the Bayesian Regularization training algorithm.

2.4 Fitting with Unimodal Approximation Optimization

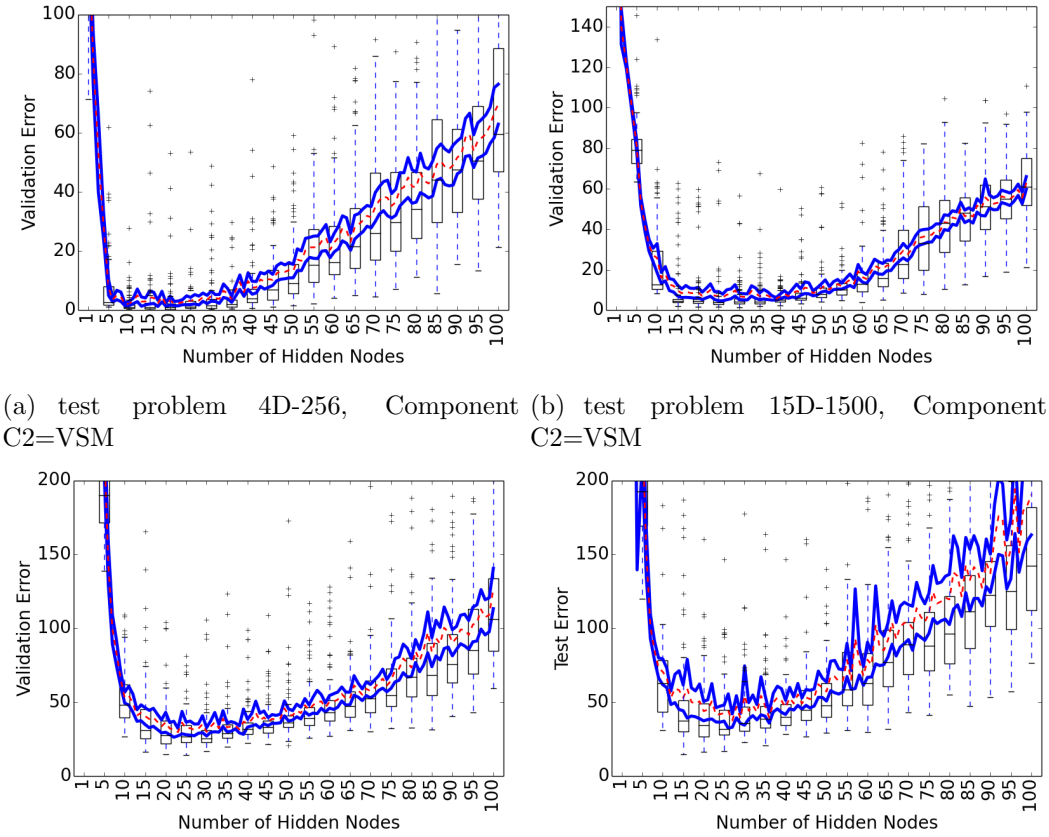
In this Section the Unimodal Approximation Optimization (UAO) algorithm is first developed. Then, the best combination of hyperparameter optimization algorithm (Component C3) and error estimation method (Component C2) is determined when Components C1 and C4 are fixed to training FFNN1s with the Early Stopping method (C1=ES) using an ensemble of the single best-trained FFNN1 (C4=1). The eight algorithms tested here are therefore $Fit(ES, C2, C3, 1)$, with C2=UAO, BO, G, or R, and C3=TSM or VSM, and it is shown that Fitting via UAO (FUA), given by $Fit(ES, VSM, UAO, 1)$, outperforms the other seven fitting methods.

Each fitting algorithm Fit is evaluated on datasets drawn from the three control problems in Section 2.5. This evaluation consists of three steps: First create a test problem CP-L, where CP is a control problem and L is the number of samples, by drawing a dataset $\mathcal{D} = \{(s_i, J_{CP}^{T-1}(s_i)) : s_i \in \mathcal{S}_{CP}^{T-1}\}_{i=1}^{L}$ from the state space $\mathcal{S}_{CP}^{T-1}$ and future value function J_{CP}^{T-1} of the second-to-last stage $T-1$ of control problem CP. For each of the three control problems in Section 2.5 three samples of different sizes are drawn, resulting in the following nine test problems:

- 4D-16, 4D-81, 4D-256. 4D is the 4-dimensional hydropower control problem
- 12D-750, 12D-1500, 12D-3000. 12D is the 12-dimensional hydropower control problem
- 15D-750, 15D-1500, 15D-3000. 15D is the 15-dimensional inventory control problem

In all cases the samples are drawn from the state space using the quasi-random

Sobol sequence [9]. Second, use Fit to train FFNN1s on $\mathcal{D}$ to obtain the FFNN1 approximation F of J_{CP}^{T-1}. And third, evaluate F using the estimated error $\widehat{GE}$ in Equation (2.3) and the L_2 error GE in Equation (2.2). Though it cannot be computed in practice, the L_2 error is accurately calculated here using Monte Carlo integration.



(a) test problem 4D-256, Component C2=VSM

(b) test problem 15D-1500, Component C2=VSM

(c) test problem 12D-1500, Component C2=VSM

(d) test problem 12D-1500, Component C2=TSM

Figure 2.1: Distribution of estimated generalization errors $X_h^{\mathcal{D},C2}$ when training an FFNN1 with h hidden nodes and using Early Stopping. The solid lines show the 95% confidence interval of the estimated mean on the domain $\mathcal{H} = \{1, 2, ..., 100\}$, and the dashed line shows the mean. The boxplots show the distribution for $h = 1, 5, 10, ..., 100$.

Each combination of test problem CP-L and error estimation method C2 creates a unique hyperparameter optimization problem in Equation (2.6) to which the optimization algorithms are applied, yielding 18 hyperparameter optimization

test problems. The distribution of $X_h^{\mathcal{D},C2}$ in Equation (2.5) can be visualized for each of these 18 problems by training 100 FFNN1s on each number of hidden nodes h to approximate the distribution of $X_h^{\mathcal{D},C2}$. Figure 2.1 shows four of these plots. The x-axis is the candidate number of hidden nodes $\mathcal{H} = \{1, 2, 3, ..., 100\}$, and the solid and dashed lines show the 95% confidence interval of the mean and the mean, respectively. The boxplots show the distributions of estimated error for $h = 1, 5, 10, ..., 100$.

Based on Figure 2.1 and the other 14 plots in Appendix 2.A, we develop the new surrogate-based Unimodal Approximation Optimization algorithm to apply in FFNN1 fitting. Surrogate optimization algorithms can often find a better solution with fewer evaluations than other algorithms by modeling the objective function with a surrogate based on the previous samples and using this information to guide future search [18, 31].

UAO uses a unimodal surrogate to estimate the objective function. A unimodal function can be drawn that is contained within the 95% confidence intervals shown in solid lines in 15 of these 18 plots, and this increases to all 18 plots when using the 95% confidence intervals. This provides evidence that the expected estimated error, which is the objective function, is unimodal. While there is no general theory describing the distribution of errors as a function of the number of hidden nodes, there is a result showing that the L_2 error of an FFNN1 trained to the globally optimal parameters, meaning Early Stopping is not used, is bounded by the unimodal function $O(H) + O(1/H)$ [1].

An important note is that the data may not be convex, which is why the more general unimodal functions are used. A convex function can be fit within the 95% confidence intervals on only 1 of the 18 plots, and at the 99% confidence interval

this increases to 14 of the 18 plots.

Based on this evidence the set of unimodal functions is sufficiently general that they can fit the objective functions. But on the other hand, making the assumption that the objective function is unimodal will result in a more accurate approximation when only a small number of evaluations have been made, i.e. number of FFNN1s that have been trained. In particular, this contrasts with Bayesian Optimization, which models the objective function with a Gaussian Process (GP) [23]. While GPs can also exactly fit these experimental objective functions, they make fewer assumptions about the shape of the function and so may not be as accurate with a small number of evaluations. Additionally, GPs assume that the noise distribution is normal, which these plots show to be false.

2.4.1 Unimodal Approximation Optimization Algorithm

The most general UAO algorithm that can be applied to more general FFNNs with multiple discrete hyperparameters (such as FFNNs with multiple hidden layers) is described here. The domain of the optimization problem is a d-dimensional finite grid $\mathcal{H}$, in Definition (1).

Definition 1. *A set $\mathcal{H}$ is a d-dimensional finite grid if it is the Cartesian product $\mathcal{H} = \{1, 2, ..., N_1\} \times \cdots \times \{1, 2, ..., N_d\}$ for d positive integers $N_1, ..., N_d$.*

The noisy objective function $\mu : \mathcal{H} \to \mathbb{R}$ must be strictly unimodal, in Definition (3).

Definition 2. *Let $\mathcal{H}$ be a d-dimensional finite grid and select $i, j, h \in \mathcal{H}$ with $h = (h_1, ..., h_d), i = (i_1, .., i_d)$, and $j = (j_1, ..., j_d)$. Then $\preceq_h$ is the unimodal partial order with minimum h if the relation $i \preceq_h j$ is satisfied if and only if for all $k = 1, ..., d$ either $h_k \leq i_k \leq j_k$ or $j_k \leq i_k \leq h_k$.*

Definition 3. *Let $\mathcal{H}$ be a d-dimensional finite grid, select $h \in \mathcal{H}$, and let $\preceq_h$ be the unimodal partial order on $\mathcal{H}$ with minimum h. A function $z : \mathcal{H} \to \mathbb{R}$ is (strictly) unimodal with minimum h if for all $i, j \in \mathcal{H}$ the relation $i \preceq_h j$ implies $(z(i) < z(j))$ $z(i) \leq z(j)$.*

Notice that $h \preceq_h i$ for all $h, i \in \mathcal{H}$, so a strictly unimodal function with minimum h is uniquely minimized at h. For $h \in \mathcal{H}$ the objective function noise is a random variable X_h, and the mean $\mathbb{E}(X_h)$ is equal to the objective function at h, $\mu(h)$. It is only assumed that each X_h has compact support. Under these constraints the optimization problem is

$$\min_{h \in \mathcal{H}} \mu(h). \tag{2.7}$$

Relating this back to the hyperparameter optimization problem, notice that the optimization problem in Equation (2.7) is the same as Equation (2.6). $\mathcal{H}$ is the candidate number of hidden nodes and X_h is the distribution of estimated errors when training an FFNN1 with h hidden nodes.

To describe the UAO algorithm the following functions are defined, where the superscript k denotes the algorithm iteration. Let $\mathbb{1}^k(h)$ for $h \in \mathcal{H}$ be a Boolean indicator that at iteration k of the UAO algorithm a sample is drawn from the distribution of X_h, and if $\mathbb{1}^k(h) = 1$ set this sampled value to $x^k(h)$; if $\mathbb{1}^k(h) = 0$ then set $x^k(h) = 0$. Define the count $n^k(h) = \sum_{i=1}^{k} \mathbb{1}^i(h)$. At iteration k if $n^k(h) > 0$ then the empirical mean $\mu^k(h)$ of X_h is

$$\mu^k(h) = \frac{1}{n^k(h)} \sum_{i=1}^{k} x^i(h). \tag{2.8}$$

Next define $\mathcal{X}_h^k = \{x^i(h) : \mathbb{1}^i(h) = 1, i = 1, ..., k\}$; let $var(\mathcal{X}_h^k)$ be the unbiased sample variance of $\mathcal{X}_h^k$; set $\mathcal{H}_m^k = \{h \in \mathcal{H} : n^k(h) \geq m\}$ to be the elements of $\mathcal{H}$ which have been selected at least m times; and select $\varepsilon > 0$. The inverse variance of this estimate can then be defined as

$$\sigma^k(h) = \begin{cases} 0 & \text{if} \quad n^k(h) = 0 \\ \frac{n^k(h)-1}{var(\mathcal{X}_h^k)+\varepsilon} & \text{if} \quad n^k(h) > 1 \\ 1 & \text{if} \quad n^k(h) = 1, |\mathcal{H}_2^k| = 0 \\ \left[\frac{1}{|\mathcal{H}_2^k|} \sum_{h' \in \mathcal{H}_2^k} var(\mathcal{X}_{h'}^k) + \varepsilon\right]^{-1} & \text{if} \quad n^k(h) = 1, |\mathcal{H}_2^k| > 0 \end{cases} \tag{2.9}$$

If $n^k(h) = 1$, it is assumed that the variances of X_h for each $h \in \mathcal{H}$ are approximately the same, so in line 4 of Equation (2.9) if $|\mathcal{H}_2^k| > 0$ then $\sigma^k(h)$ is set to the inverse mean of the estimated variances. Otherwise, in line 3 $\sigma^k(h)$ is set to 1. A small constant ε is added in the denominator both to avoid the potential issue that $var(\mathcal{X}_h^k) = 0$ and to aid the proof of convergence.

To construct the unimodal surrogate, the weighted quadratic penalty U of fitting a unimodal function z with minimum h to the data μ^k and σ^k for each $h \in \mathcal{H}$ is first calculated as

$$U(h, \mu^k, \sigma^k) = \min_{z(h)} \sum_{h \in \mathcal{H}} \sigma^k(h)(z(h) - \mu^k(h))^2 \tag{2.10}$$
$$\text{s.t. } z(i) \leq z(j) \text{ if } i \preceq_h j$$

The constraints enforce that z is unimodal with minimum h. The set of points h in $\mathcal{H}$ defined by

$$\mathcal{U}(\mu^k, \sigma^k) = \left\{ h \in \mathcal{H} : U(h, \mu^k, \sigma^k) = \min_{h' \in \mathcal{H}} U(h', \mu^k, \sigma^k) \right\} \tag{2.11}$$

are those for which a unimodal function with minimum h can best fit the data. At the next iteration of the UAO algorithm $k+1$ it is reasonable to sample X_h for some element h either in or near to $\mathcal{U}(\mu^k, \sigma^k)$. Optimal algorithms have been developed for calculating the set $\mathcal{U}(\mu^k, \sigma^k)$ [35].

The Unimodal Approximation Optimization algorithm is presented in Algorithm 1. In Lines 1 and 2 the constants ε and I are defined, and the domain $\mathcal{H}$ and noisy objective function are selected. The algorithm is initialized in Lines 3 and 4 by selecting K_{init} elements $h_1, ..., h_{K_{init}} \in \mathcal{H}$, sampling X_{h_i}, and updating $x^i(h_i)$ and $\mathbb{1}^i(h_i)$ for $i = 1, ..., K_{init}$. Lines 6-8 specify how to select the next element h. Line 7 helps improve the exploration characteristics of the algorithm. Instead of selecting any $\tilde{h} \in \mathcal{U}(\mu^k, \sigma^k)$, $\tilde{h}$ is uniformly randomly selected from the subset of $\mathcal{U}(\mu^k, \sigma^k)$ that maximizes the Euclidean L_1 distance $||\cdot||_1$ to any $h' \in \mathcal{H}_1^k$; that is, $\tilde{h}$ is selected as far from any h' such that $n^k(h') > 0$ as possible. The reason why this helps improve the exploration is explained in Lemma 1, which shows that when the UAO iteration k is small with respect to the number of elements in $\mathcal{H}$ the

Algorithm 1 Unimodal Approximation Optimization (for Fitting FFNN1s)

1: Select $I \geq 1$; $\varepsilon > 0$; d-dimensional finite grid $\mathcal{H}$ ($d = 1$, $\mathcal{H}$ is candidate number of hidden nodes)
2: Let X_h be noise distributions, $\mu(h) = \mathbb{E}(X_h), h \in \mathcal{H}$ be objective function ($\mu(h)$ is expected estimated error when training an FFNN1 with h hidden nodes)
3: Initial data $x^k(h)$ and $\mathbb{1}^k(h)$ for $h \in \mathcal{H}, k = 1, ..., K_{init}$ (train K_{init} FFNN1s)
4: Set $k \leftarrow K_{init}$; $x_{best} \leftarrow$ smallest sampled $x^k(h)$ from Step (3); $F \leftarrow$ best-trained FFNN1
5: **while** Condition **do**
6: Compute $\mathcal{U}(\mu^k, \sigma^k)$ using Equations (2.10), (2.11)
7: Uniformly randomly select $\tilde{h} \in \underset{h \in \mathcal{U}(\mu^k, \sigma^k)}{\operatorname{argmax}} \left(\min_{i \in \mathcal{H}_1^k} ||h - i||_1 \right)$
8: Uniformly randomly select $h \in \{i : ||\tilde{h} - i||_1 \leq I, i \in \mathcal{H}\}$
9: Draw sample $x^k(h)$ from X_h (train FFNN1 with h hidden nodes, $x^k(h)$ is estimated error)
10: **If** $x^k(h) < x_{best}$ **then** $x_{best} \leftarrow x^k(h)$, $F \leftarrow F^k$
11: Update $k \leftarrow k + 1$, x^k, $\mathbb{1}^k$ **end**
12: **end while**
13: **return** $\mathcal{U}(\mu^k, \sigma^k)$, ($F$ is best-trained FFNN1)

set $\mathcal{U}(\mu^k, \sigma^k)$ is potentially large. In Line 8 the selection $\tilde{h}$ is randomly perturbed using input I. This can be viewed as a sort of annealing parameter, though it is also critical to prove almost sure convergence. Finally, the algorithm is iterated until some termination Condition in Line 5 is satisfied, such as continuing for a predefined number of allowed iterations or until a time budget has expired.

2.4.2 UAO Theoretical Considerations and Convergence

All proofs are presented in Appendix 2.B. Lemma 1 below implies that when the UAO algorithm has only been repeated for a small number of iterations then $\mathcal{U}(\mu^k, \sigma^k)$ can be large. For example, if $d = 1$ and at iteration k $n^k(h) = 0$ for $h = i, .., j$, then $U(h, \mu^k, \sigma^k) = U(i, \mu^k, \sigma^k)$ for $h = i, ..., j$.

Lemma 1. *Select k and $h, H \in \mathcal{H}$ such that at the k^{th} iteration of Algorithm 1 $n^k(h) = n^k(H) = 0$, and let $\preceq_h$ and $\preceq_H$ be the unimodal partial orders with minimum h and H (Definition 3). If $i \preceq_h j$ iff $i \preceq_H j$ for all $i, j \in \mathcal{H}$ such that $n^k(i) > 0$ and $n^k(j) > 0$, then $U(h, \mu^k, \sigma^k) = U(H, \mu^k, \sigma^k)$.*

The main theoretical result is Theorem 1, which states that under two assumptions the UAO algorithm converges almost surely. The primary assumption is that the mean objective function μ is strictly unimodal with some unique minimum h^*.

The second assumption is that the noise X_h has compact support for each $h \in \mathcal{H}$. In the FFNN1 fitting process this condition requires that the distribution of estimated generalization errors when trained with h hidden nodes has compact support. The probability is zero that the estimated error is less than zero since the errors are non-negative. For any dataset $\tilde{\mathcal{D}}$ the estimated generalization error $\widehat{GE}(F(\cdot; \theta, h); \tilde{\mathcal{D}})$ is bounded above if the parameters θ of the FFNN1 F are all bounded. Lemma 3 in Appendix 2.B proves that if the Levenberg-Marquardt training algorithm is used, the initialized parameters θ are bounded, and the Levenberg-Marquardt algorithm is terminated after a finite number of iterations then the trained parameter values of θ are also bounded.

Theorem 1. *Set $I \geq 1$ and $\varepsilon > 0$ in the UAO algorithm (Algorithm 1). Select a d-dimensional finite grid $\mathcal{H}$, and for each $h \in \mathcal{H}$ let X_h be a random variable*

with mean $\mu(h) = \mathbb{E}(X_h)$ *and compact support. Assume the function* $\mu : \mathcal{H} \to \mathbb{R}$ *is strictly unimodal with unique minimum* h^*.

Let $\mathcal{U}^1, \mathcal{U}^2, ...$ *be the infinite sequence of random elements with probability measure* P *such that at the* k^{th} *iteration of the while loop (line* 5 *of Algorithm* 1*)* $P(\mathcal{U}^k = \mathcal{U})$ *is the probability that the solution in Equation (2.11) is* $\mathcal{U} \in 2^{\mathcal{H}} \backslash \emptyset$ *conditioned on the initial data in Line* 3 *of the UAO algorithm. Then*

$$P(\lim_{k \to \infty} \mathcal{U}^k = \{h^*\}) = 1,$$

i.e. the UAO algorithm converges almost surely to the unique global minimum.

2.4.3 Comparison of Hyperparameter Optimization Algorithms and Error Estimation Methods

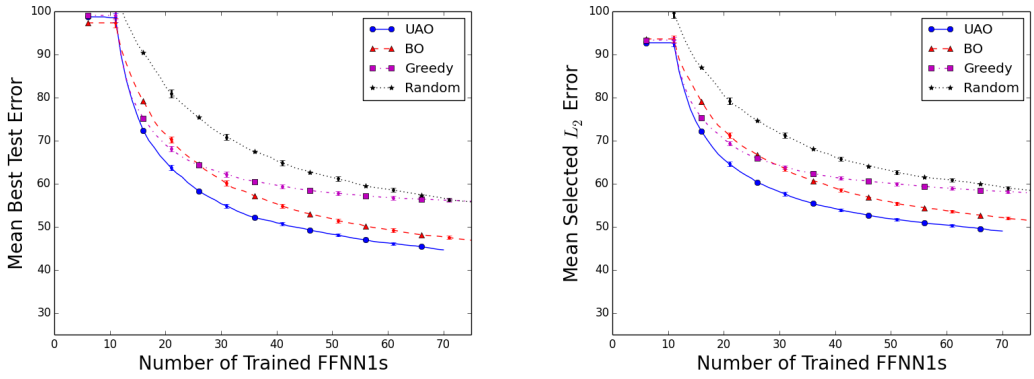
Again, there are eight FFNN1 fitting algorithms $Fit(C1 = ES, C2, C3, C4 = 1)$, and they are tested on 18 test problems. The three C3 optimization algorithms compared to UAO are:

a) Bayesian Optimization This is the BO algorithm as described in [34]. The posterior distribution is created using the Matèrn 5\2 kernel, the length scale and noise hyperparameters are numerically integrated out, and the Expected Improvement criterion is optimized. This algorithm is initialized with K_{init} trained FFNN1s.

b) Greedy This algorithm has already been applied to FFNN1 hyperparameter optimization [33]. K_{init} FFNN1s are initially trained with a predefined number of hidden nodes, and H is selected as the number of hidden nodes that yielded the smallest experimental estimated generalization error. All other FFNN1s are then trained with H hidden nodes.

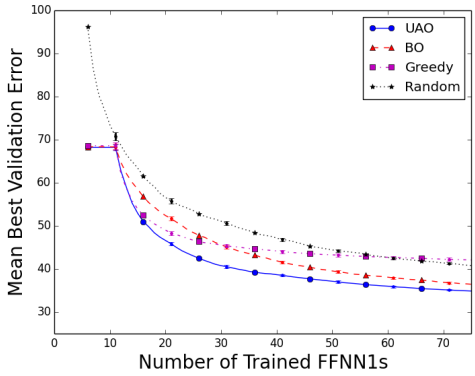
c) Random The number of hidden nodes is uniformly randomly selected.

These algorithms were all run with the same parameters. The domain of the optimization problem, which is the candidate number of hidden nodes, is $\mathcal{H} = \{1, 2, ..., 100\}$. The initialization method for UAO, BO, and the Greedy algorithms is to train $K_{init} = 11$ FFNN1s, where one FFNN1 was trained with each of $h = 1, 10, 20, ..., 100$ hidden nodes. It was found that UAO is insensitive for small values of parameters I and ε, and these are set to $I = 1$ and $\varepsilon = 10^{-3}$.

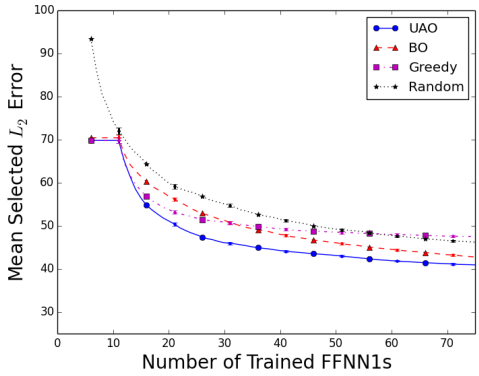


(a) best test error

(b) L_2 error of FFNN1 with best test error



(c) best validation error



(d) L_2 error of FFNN1 with best validation error

Figure 2.2: Progress plots comparing optimization algorithms applied to dataset 12D-750, i.e. the dataset from the 12D Hydropower problem with $L = 750$; top plots use Component C2=TSM, bottom use C2=VSM. Plots show mean performance over $1,000$ trials, errorbars show standard deviation of estimate of the mean and are often too small to be seen.

Each of the eight fitting algorithms can be compared using progress plots that measure either estimated error or L_2 error. In these plots the x-axis is the number of trained FFNN1s. If the comparison uses the estimated error, then the y-axis is the minimum of the estimated errors of the FFNN1s that have been trained so far. This plot is non-increasing. If the comparison uses the L_2 error, then the y-axis is the L_2 error of the FFNN1 that has been selected based on the estimated error. Because the estimated error is not completely accurate this plot is not necessarily non-increasing. The algorithms can be compared using either performance metric

by determining which performance plot decreases the fastest. Each algorithm is run $1,000$ times in order to obtain the mean and standard deviation of these performances.

Figure 2.2 shows these comparisons using the 12D-750 test dataset (i.e. fitting the value function of the 12 dimensional Hydropower problem with $L = 750$ samples). The top plots use the Test Set Method, the bottom plots use the Validation Set Method, the left plots show the estimated error performance, and the right plots show the corresponding L_2 error performance. It can be seen in all four plots that the UAO algorithm in outperforms others. Comparing the figures on the left to those on the right show that both the validation and test errors tend to underestimate the L_2 generalization error. Also, comparing plot (b) to (d) shows that a smaller L_2 error is achieved when the validation set is used to estimate the generalization error. The progress plots for all 9 datasets are shown in Appendix 2.C.

A more complete analysis on all 9 test problems, shown in Tables 2.4.3 and 2.2, uses speedup to compare pairs of algorithms. The speedup $S^{error}_{A,B}(t)$ is a ratio of how long it takes algorithm B to achieve the same performance that algorithm A achieved after t seconds of computation time; $S^{error}_{A,B}(t) > 1$ means algorithm A is faster and $S^{error}_{A,B}(t) < 1$ means algorithm B is faster.

Figure 2.3 provides an intuitive explanation of speedup, and a rigorous definition is in Appendix 2.D. This is a sketch of a progress plot like those in Figure 2.2, except the x-axis is computation time. The y-axis is $error$, which can be the estimated or L_2 error. At time t_0 the performances are not statistically different so, calling the two algorithms *solid* and *dashed*, $S^{error}_{solid,dashed}(t_0) = S^{error}_{dashed,solid}(t_0) = 1$. The earliest time the performance of *solid* is not statistically different from the per-

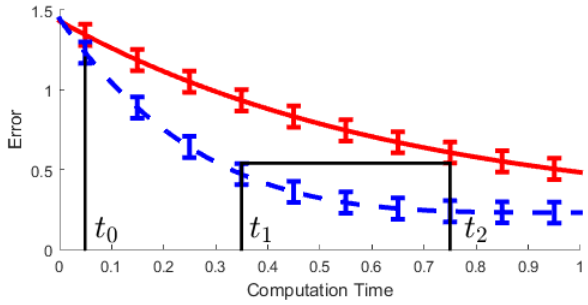


Figure 2.3: Visual explanation of speedup comparison $S^{error}_{A,B}(t)$. $S^{error}_{solid,dashed}(t_0) = S^{error}_{dashed,solid}(t_0) = 1$ (algorithms same), $S^{error}_{dashed,solid}(t_1) = t_2/t_1 > 1$ (dashed is better), $S^{error}_{solid,dashed}(t_2) = t_1/t_2 < 1$ (solid is worse)

formance of *dashed* at time t_1 is at time t_2. Therefore, $S^{error}_{dashed,solid}(t_1) = t_2/t_1 > 1$, which indicates that *solid* requires more time to achieve a similar performance to the performance achieved by *dashed* at time t_1. Conversely $S^{error}_{solid,dashed}(t_2) = t_1/t_2 < 1$, which indicates that *dashed* requires less time to achieve a similar performance to the performance achieved by *solid* at time t_2.

The speedups presented in Table 2.4.3 compare fitting algorithms $Fit(ES, C2, C3, 1)$ to $Fit(ES, C2, UAO, 1)$ and show that UAO is almost always faster than the other algorithms. The speedup is calculated using the validation error (left three columns) when C2=VSM and the test error when C2=TSM. The speedups presented in Table 2.2 compare fitting algorithms $Fit(ES, VSM, C3, 1)$ to $Fit(ES, TSM, C3, 1)$ and show that it is always better to use the Validation Set Method C2=VSM. The speedup is calculated using the L_2 error.

		Optimize Validation Error, C2=VSM[b]						Optimize Test Error, C2=TSM[b]					
4D[c] (Hydropower)	L^c	16		81		256		16		81		256	
	time t [seconds]	12	25	17	35	40	80	12	25	17	35	40	80
	B0[a]	**2.00**	**> 10**	**1.47**	**1.43**	**1.08**	**1.02**	**0.58**	**0.68**	**1.24**	**1.26**	**1.08**	**1.09**
	Greedy[a]	**> 9**	**> 5**	**> 6**	**> 3**	**> 12**	**> 8**	**2.00**	**> 5**	**> 8**	**> 4**	**2.25**	**> 8**
	Random[a]	**2.50**	**2.52**	**1.12**	**1.46**	**1.33**	**1.38**	**1.08**	**1**	**1.41**	**1.77**	**1.58**	**1.75**
12D[c] (Hydropower)	L^c	750		1500		3000		750		1500		3000	
	time t [seconds]	30	60	60	120	90	180	30	60	60	120	90	180
	B0[a]	**1.23**	**1.16**	**1.50**	**2.50**	**1**	**0.87**	**1.20**	**1.07**	**1.55**	**1.70**	**1**	**1**
	Greedy[a]	**1.36**	**> 10**	**1**	**1**	**1**	**1**	**1.36**	**> 10**	**1.28**	**> 10**	**1**	**1**
	Random[a]	**2.13**	**2.32**	**1.76**	**2.17**	**1.08**	**1.21**	**2.13**	**2.35**	**1.46**	**1.55**	**1.02**	**1.21**
15D[c] (Inventory)	L^c	750		1500		3000		750		1500		3000	
	time t [seconds]	40	80	200	400	500	1000	40	80	200	400	500	1000
	B0[a]	**1.35**	**1.06**	**1.61**	**2.18**	**1**	**1**	**1.88**	**1.86**	**1.40**	**1.29**	**1**	**1**
	Greedy[a]	**> 14**	**> 7**	**1**	**1**	**1**	**1**	**1.45**	**> 7**	**1.07**	**1.09**	**1**	**1**
	Random[a]	**2.10**	**2.08**	**1.75**	**1.82**	**1**	**1**	**3.02**	**3.25**	**1.55**	**1.51**	**1**	**1**

Table 2.1: Component 3 speedup results — values greater than one means UAO is faster. Speedups $S^{error}_{UAO\text{-}C2,C3\text{-}C2}(t)$ shown in bold boxes compare $Fit(ES, C2, C3, 1)$, with Component C2 denoted by superscript (b) and C3 denoted by (a), to $Fit(ES, C2, UAO, 1)$ on 9 test problems denoted by c. If C2=TSM then *error* is test set error, if C2=VSM then *error* is validation set error. For example, consider the top left bolded 3X2 box, where the number in the first row and first column is 2.00. This indicates that on test problem 4D-16 (4D is on the left side of the row, L=16 is on the top of the column) the fitting algorithm $Fit(ES, VSM, BO, 1)$ (BO is on the left side of the row, VSM is on the top of the column) required $2.00 * 12 = 24$ seconds of computation time (where $t = 12$ seconds is on the top of the row) to obtain a mean estimated error not statistically larger than what the fitting algorithm $Fit(ES, VSM, UAO, 1)$ obtained in 12 seconds.

	4D Hydropower[c]			12D Hydropower[c]			15D Inventory[c]		
L[c]	16	81	256	750	1500	3000	750	1500	3000
time t [seconds]	25	35	80	60	120	180	80	400	1000
UAO[a]	> 2	1.14	> 2	> 2	> 2	3.16	1.21	> 2	> 2
BO[a]	0.56	1.37	> 2	2.63	3.55	3.84	1.58	> 2	> 2
Greedy[a]	> 4	> 4	> 3	> 10	> 10	5.17	> 7	> 11	> 10
Random[a]	1.36	3.34	4.36	3.50	4.28	3.45	2.80	3.38	2.15

Table 2.2: Component 2 speedup results — values greater than one means VSM is better than TSM. Speedups $S^{L_2}_{C3\text{-}VSM,C3\text{-}TSM}(t)$ compare $Fit(ES, VSM, C3, 1)$, with Component C3 denoted by superscript (a), to $Fit(ES, TSM, C3, 1)$ on 9 test problems denoted by c. Comparison uses L_2 error of trained FFNN1 with smallest estimated error. For example, consider the top left number in the table, which is > 2. This indicates that in the test problem 4D-16 (4D and $L = 16$ are both on top of the row) the fitting algorithm $Fit(ES, TSM, UAO, 1)$ (algorithm UAO is on the left side of the row) required $(> 2) * 25 = (> 50)$ seconds of computation time ($t = 25$ seconds is shown on the top of the column) to obtain a mean L_2 error not statistically larger than what the fitting algorithm $Fit(ES, VSM, UAO, 1)$ obtained in $t = 25$ seconds.

2.5 Test Problems

The impact of using different FFNN1 training methods to approximate the future value function are compared on two hydropower problems and an inventory problem. To help examine if the impact is a function of the number of state space dimensions there is a hydropower problem with 4 state dimensions, a hydropower problem with 12 state dimensions, and an inventory problem with 15 state dimensions.

2.5.1 4D Hydropower

This test problem has four state space dimensions and is presented in Section 6 of [22], Equations (9) through (11). The discretized stochastic formulation is used, where each of the two independent lognormal inflows are replaced with a discrete three-sample approximation.

2.5.2 12D Hydropower

This test problem has twelve state space dimensions and is representative of a large hydropower system. The objective is to determine how much water to release from the reservoirs (hydropower units) in the network shown in Figure 4.1. The index used to reference each reservoir is shown inside the reservoir symbol. The gray reservoirs have a storage capcity, and the white reservoirs are 'run of river' which are used to represent time delays and so have no storage. It takes six time steps for water to travel from reservoir 3 to 6, so the problem is formulated with a horizon of 8 time steps to ensure water traverses the whole reservoir system. Note that in

this Subsection subscripts denote elements of a vector.

State Space and Decision Variables

The 12-dimensional state space $\mathcal{S}$ represents the total amount of water in each reservoir at the beginning of time step k, with s_i^k the amount of water in reservoir i at stage k. The six decision variables $R_i^k, i = 1, ..., 6$ are the total amount of water to release from reservoirs 1 through 6 and are constrained by

$$0 \leq R_i^k \leq s_i^k, i = 1, ..., 6. \tag{2.12}$$

Inflows

The inflows into reservoirs $1, 2$ and 3 are mutually independent and time independent. The stochastic inflows are discretized approximations of a lognormal distribution with means $\mu_1 = 2, \mu_2 = 3, \mu_3 = 4$ and variances $\sigma_1^2 = 1^2, \sigma_2^2 = 1.5^2, \sigma_3^2 = 2^2$. The approximations are created by assigning probabilities of 1/6, 2/3, and 1/6 to the 5^{th}, 50^{th}, and 95^{th} percentiles. The expectation over these inflows is computed with the $3^3 = 27$ discrete samples.

State Dynamics

The six storage reservoirs 1 through 6 have a maximum capacity $W_{i=1,...,6} = (10, 15, 20, 20, 16, 27)$. Using the same model as in [7], a floodway is used to spill water that exceeds this capacity. Let $Upstream_i \subset \{1, 2, ..., 12\}$ be the set of reservoirs that release water immediately upstream of reservoir i. For $i \in \{1, 2, 3\}$ let q_i^k be the stochastic inflow into reservoir i, and otherwise set $q_i^k = 0$. The state

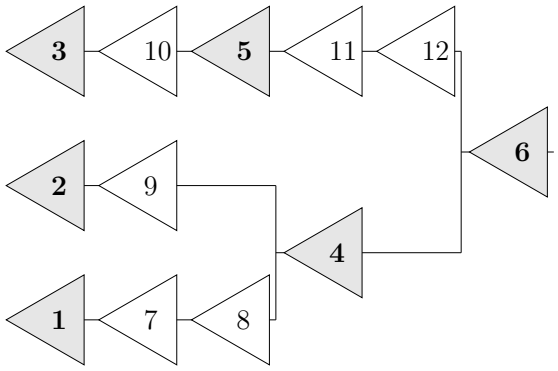


Figure 2.4: Reservoir network diagram. Numbers indicate the reservoir number. Gray reservoirs have water capacity and white reservoirs are run-of-the-river. Water enters the system through reservoirs 1, 2, and 3.

dynamics equation is

$$s_i^{k+1} = \min\left\{s_i^k + q_i^k - R_i^k + \sum_{j \in Upstream_i} R_j^k, W_i\right\}. \tag{2.13}$$

Cost Function

The cost function depends on the stage, thereby making this a nonstationary problem. The terminal value function is

$$C^T(s) = \sum_{i=1}^{6} (s_i^k - \hat{W}_i)^2 \tag{2.14}$$

with $\hat{W} = (5, 10, 10, 15, 12, 20)$. This cost represents a penalty term for deviations from a target water volume level of $\hat{W}$ in the storage reservoirs. The stage cost

$$C^k(R^k) = \sum_{i=1}^{6} w_i \Big(\alpha^k Q(\max\{0, R_i^k - \hat{R}_i\}) + \beta^k Q(\max\{0, \hat{R}_i^k - R_i\})\Big) - \sum_{i=1}^{6} w_i \log(R_i^k + 1) \tag{2.15}$$

is the sum of a penalty for not releasing a target amount of water (first summation) and a benefit for releasing more to produce hydropower (second summation). The

penalty is given by a smoothed 'V' shape

$$Q(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ \frac{z^3}{4} - \frac{z^4}{16} & \text{if } 0 < z \leq 2 \\ z - 1 & \text{if } z \geq 2 \end{cases} \tag{2.16}$$

The target releases are $\hat{R} = (2, 3, 4, 5, 4, 9)$. The weights $w = (1, 1, 1, 1.2, 1.2, 1.4)$, cost of over-production $\alpha = (1, 2, 3, 4, 4, 4, 4, 4)$, and cost of under-production $\beta = (1, 1, 1, 1, 2, 3, 4, 4)$ are stage dependent. The benefit function uses the same weights w_i as previously.

2.5.3 15D Inventory

This is a common inventory test problem that has been used multiple times to evaluate an ADP solution. For a full description, see [13]. We use a variation with 5 inventory products, resulting in a state space with 15 dimensions, and use a finite horizon formulation with 4 stages.

2.6 Comparing FFNN1 Fitting Algorithms on ADP Test Problems

Based on the background presented in Section 2.3 and the results in Section 2.4 four fitting algorithms are tested on the three test control problems. These are:

Fitting via UAO (FUA), $Fit(ES, VSM, UAO, 1)$

This method is the result of analysis in Section 2.4. There, results showed Early Stopping (ES) with the Validation Set Method (VSM) and Unimodal Approximation Optimization (UAO) hyperparameter optimization algorithm outperformed other combinations. FUA uses an ensemble of $C4 = 1$ (i.e. the single trained FFNN1 with the smallest estimated error). The UAO algorithm uses hyperparameters $I = 1$ and $\varepsilon = 10^{-3}$, the candidate number of hidden nodes UAO searches over is $\mathcal{H} = \{1, ..., 100\}$ (Equation (2.6)), and $K_{init} = 11$ FFNN1s are trained in the initialization phase with $h = 1, 10, 20, ..., 100$ hidden nodes, and terminates after 40 FFNN1s are trained.

Ensemble FUA (EFUA), $Fit(ES, VSM, UAO, 5)$

The EFUA algorithm runs the FUA algorithm, but upon termination the 5 best FFNN1s that have been trained are selected. The value function is then the mean of these FFNN1s.

Fitting with Bayesian Regularization (FBR), $Fit(BR, none, none, 1)$

Training a single FFNN1 with Bayesian Regularization (BR) takes considerably more computation time than when using ES, so only a single FFNN1 is trained in this fitting algorithm.

Adaptive Value Function Approximation (AVFA), $Fit(ES, TSM, AVFA, 1)$

The AVFA algorithm was previously developed specifically for fitting FFNN1s in ADP, and it is used as a direct comparison [16]. AVFA is an iterative algorithm that finds both the number of state space samples L and the number of hidden nodes by training multiple FFNN1s and iteratively increasing both until the test set error falls below the preset target error. In [16] the AVFA algorithm was modified so that it instead terminates when the number of samples reached a target number in order to compare it to other algorithms that use a fixed number of samples. The same approach is used here. Note that the AVFA fitting method uses Early Stopping, the Test Set Method, and an ensemble of 1.

The two criteria used to evaluate a control problem solution are the computation time required to obtain the solution and its resulting performance. In order to perform a fair comparison of computation time all tests are run on the same hardware. The performance of the solution is measured with the mean expected cost. The expected cost C of starting in state s^1 and using the resulting solution policies $\pi^1, ..., \pi^{T-1}$ obtained by applying the ADP algorithm (Equation (2.1)) is

$$C(s^1, \pi^1, ..., \pi^{T-1}) = \underset{\omega^1,..,\omega^{T-1}}{\mathbb{E}} \Big[\sum_{k=1}^{T-1} c^k(s^k, \pi^k(s^k), \omega^k) + c^T(s^T) : s^{k+1} = g(s^k, \pi(s^k), \omega^k) \Big], \tag{2.17}$$

and the mean expected cost (MEC) $\bar{C}$ is the mean cost over the entire state space

$$\bar{C}(\pi^0, ..., \pi^{T-1}) = \frac{1}{\int_{s \in \mathcal{S}^1} ds} \int_{s \in \mathcal{S}^1} C(s, \pi^0, ..., \pi^{T-1}) ds. \tag{2.18}$$

This integral can be approximated by selecting M initial states s_i^1 and random variable sample sequences $\omega_i^1, ..., \omega_i^{T-1}$ for $i = 1, ..., M$. The MEC can be approximated

by

$$\bar{C}'(\pi^0, ..., \pi^{T-1}) \approx \frac{1}{M} \sum_{i=1}^{M} \sum_{k=0}^{T-1} c^k(s_i^k, \pi^k(s_i^k), \omega_i^k) + c^T(s_i^T), \tag{2.19}$$

and the variance of this estimate is

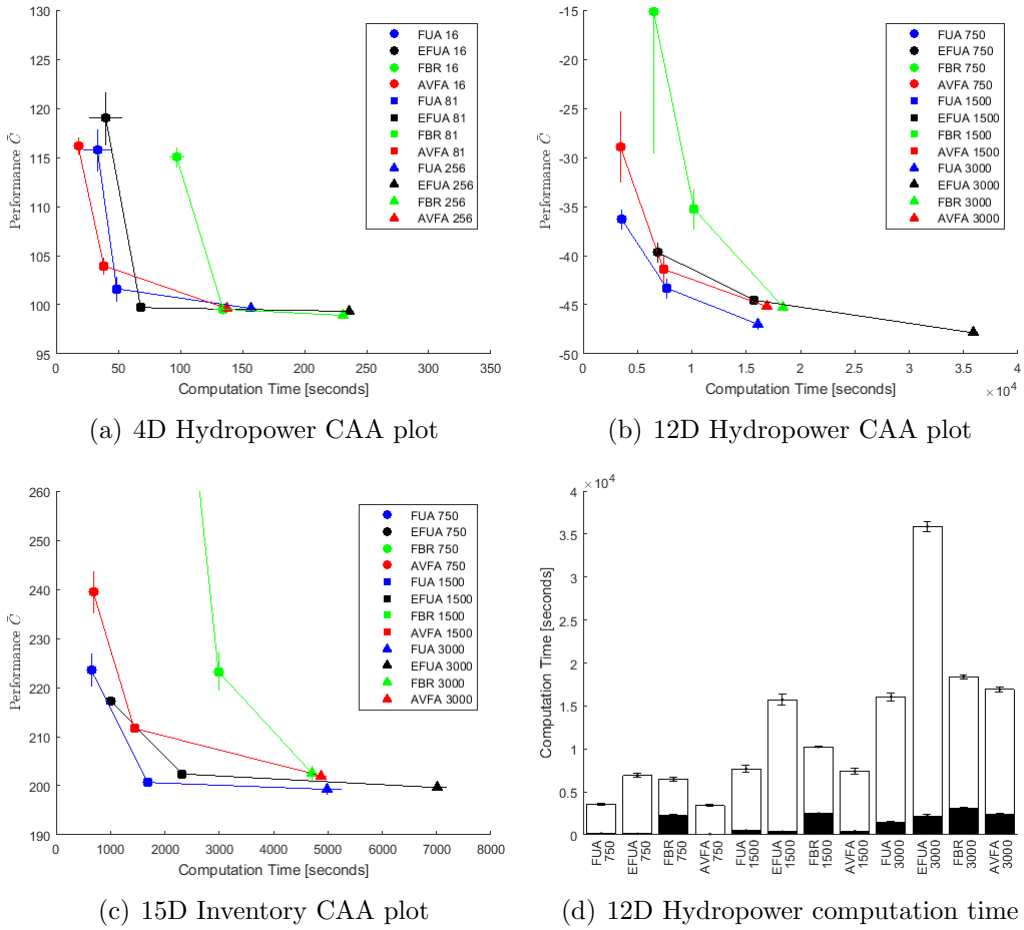
$$\sigma^2(\bar{C}') = \frac{1}{M} var\Big(\{ \sum_{k=0}^{T-1} c^k(s_i^k, a^k(s_i^k), \omega_i^k) + c^T(s_i^T) \}_{i=1}^{M} \Big), \tag{2.20}$$

where again *var* is the unbiased sample variance. The run time and MEC performance are inversely related: by evaluating more samples L to include in the dataset $\mathcal{D}$ the computation time increases but, because the FFNN1 should have a smaller generalization error, the MEC will likely decrease.

The Statistical Accuracy Analysis (SAA) and the Computation and Accuracy Analysis (CAA) assessments are designed to analyze this trade-off. These analyses are similar to those in [16], and are applied to the following data. For each test problem, each of the four FFNN1 fitting methods are used to approximate the value function with an FFNN1. For each test problem and fitting method, three values for the number of state space samples L are tested, the same as in Section 2.4. The MEC $\bar{C}'$ in Equation (2.19) and error estimate $\sigma^2(\bar{C}')$ in Equation (2.20) are then calculated. This process is repeated for five trials. Altogether, there are three control problems, four fitting methods, three values for L, and five trials, resulting in 180 runs of the ADP algorithm.

The **Computation and Accuracy Analysis** consists of plotting the trade-off between the runtime on the x-axis and the MEC performance on the y-axis, as seen in plots (a), (b) and (c) in Figure 2.5. The markers are represented by both color and shape, where markers of the same color use the same FFNN1 fitting algorithm and markers of the same shape use the same number of state space samples L. The y-coordinate is the mean of the five trials' MEC $\bar{C}'$, each as calculated in Equation

(2.19). Error bars show the variance over the five trials, although the variance in computation time results in error bars that are smaller than the markers. The trial results are shown in Appendix 2.E. Because the two goals are small runtimes and small MEC the test cases which are non-dominated, meaning there is no other test case that has both a smaller runtime and more negative cost, are Pareto optimal. The set of all the Pareto optimal test cases form the Pareto front and outperform others. Three main observations can be made from these CAA figures:



(a) 4D Hydropower CAA plot

(b) 12D Hydropower CAA plot

(c) 15D Inventory CAA plot

(d) 12D Hydropower computation time

Figure 2.5: Computation and Accuracy Analysis in plots (a)-(c). Some BR data points have MEC values too large (bad) to fit on the plot. Plot (d) shows computation time of the 12D Hydropower problem. Black bars are training time, white bars are time to evaluate state space samples, and bar height is total time.

- The FUA algorithm always yielded means over five trials, shown as the blue

markers, that are on the Pareto front in the 12D Reservoir problem and the 15D Inventory problem. This means that when compared on a runtime budget FUA yielded a better MEC than the other FFNN1 fitting algorithms. In the 4D Reservoir problem the other fitting algorithms are either on or are much closer to the Pareto front. This may imply that as the state space dimension becomes larger, e.g. 12 or 15 instead of only 4, the FFNN1 fitting method will have a larger impact on the performance.

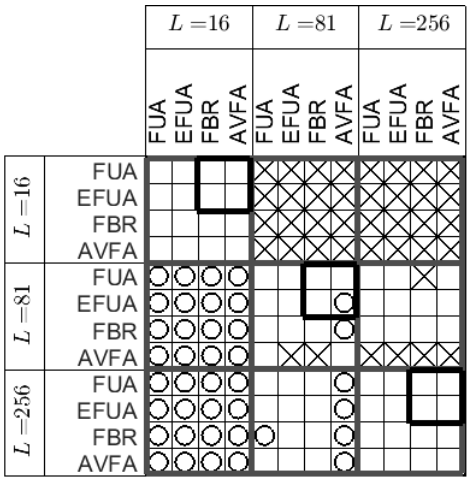
- The BR method in green does not perform as well as the other FFNN1 fitting methods. In all three test problems when the smallest number of state space samples is used most of the BR data points have an MEC that is too large to be seen in the plots. With more state space samples BR does have comparable performance.
- As expected, the ensemble EFUA method in black often has a more negative (better) MEC than other fitting algorithms for a given number of state space samples L, but the significantly increased computation time means EFUA ends up not always being near the Pareto front, i.e. other methods can yield equal performance in less time.

Plot (d) in Figure (2.5) shows the computation time breakdown of the 12D Hydropower problem. The black bars are the time spent on training FFNN1s and the white bars are the time to evaluate the future value function; the total bar height is the total computation time. The errorbars show the variance over the five trials. The analogous plots for the 4D Hydropower and 15D Inventory problem are in Appendix 2.F. The three test cases FBR-750, FBR-1500, and FBR-3000 (i.e. using the FBR fitting algorithm with $L = 750, 1500$ and 3000 samples) show that it takes significantly longer to train a single FFNN1 (black bars) using BR

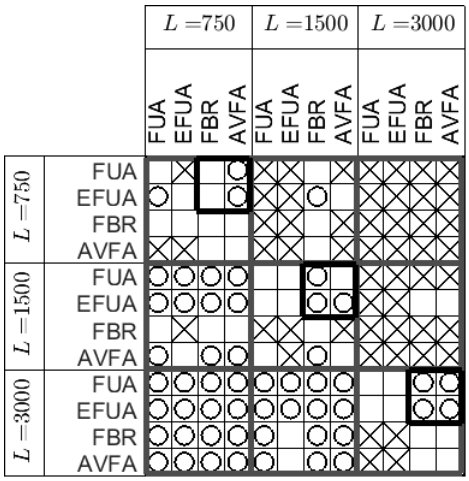
than training many FFNN1s using ES (used in FUA, EFUA, and AVFA). Also, the ensemble method EFUA takes more computation time than FUA because solving the optimization problem in Equation (2.1) requires evaluating the value function approximation (white bars) many times and it is more expensive to calculate the mean of five FFNN1s for the ensemble as opposed to evaluating a single FFNN1.

The **Statistical Accuracy Analysis** analyzes the MEC results from each combination of fitting algorithm and selection of L, which will be called 'test cases.' For every pair of test cases, SAA takes the corresponding two sets of five MEC values and tests the null hypothesis that the mean values are the same. Figure 2.6 shows which tests cases are statistically different at the 5% confidence level. An 'X' indicates the test case at the top of the column statistically outperforms the test case at the left of the row, an 'O' is the reverse, and an empty box means there is no statistically significant difference. The thick gray lines, which form a 3-by-3 grid, divide the test cases by the number of state space samples L. This helps to show the expected result that increasing L significantly improves the performance of all four fitting algorithms, as evidenced by the large number of X's in the upper-right corner.

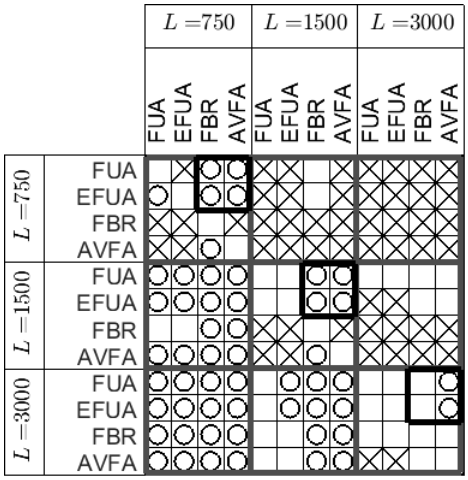
The most important results in each plot are shown in the three 2-by-2 boxes outlined in thick black lines. For a fixed number of state space samples L, these boxes compare the FUA and EFUA algorithms, labeled on the left-hand side of the rows, to the FBR and AVFA algorithms, labeled on the top of the columns. These boxes contain either empty squares or circles, which indicates that for a fixed L the FUA and EFUA fitting methods always outperform or are not statistically different from the FBR and AVFA FFNN1 fitting methods.



(a) 4D Reservoir



(b) 12D Reservoir



(c) 15D Inventory

Figure 2.6: Statistical Accuracy Analysis. An 'X' indicates the column test case statistically outperforms the row test case at the 5% level, an 'O' is the reverse, and an empty square indicates no statistical difference.

2.7 Conclusions

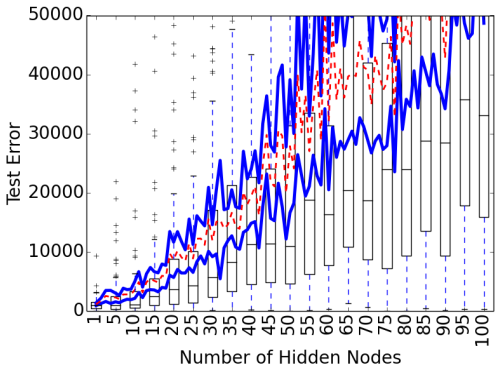
We developed a new ADP method called Fitting via Unimodal Approximation Optimization, or FUA, that we showed outperforms other methods. The FUA algorithm is applicable to general stochastic dynamic programming applications and is not limited to hydropower problems.

FUA trains multiple FFNN1s and optimizes the number of hidden nodes via the Unimodal Approximation Optimization (UAO) algorithm we developed. It was demonstrated that UAO outperforms other optimization algorithms including Bayesian Optimization on fitting the number of hidden nodes. Although in this work UAO was only numerically applied to the problem of optimizing the single FFNN1 hyperparameter, it could be applied to optimizing the multiple hyperparameters of more general FFNNs. Significantly, we proved UAO converges almost surely to the optimal solution when the noisy objective function is unimodal.

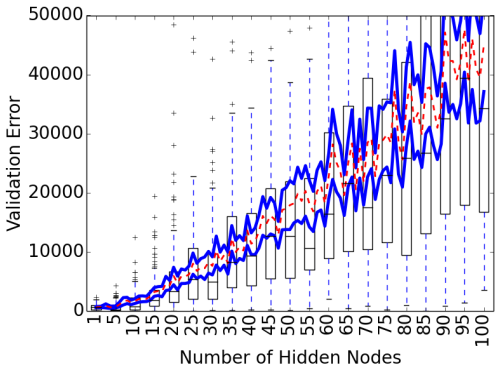
We developed the Statistical Accuracy Analysis (SAA) and Computation and Accuracy Analysis (CAA) tests to compare the relative accuracy and computational efficiency of stochastic dynamic programming solutions calculated using different algorithms. FUA was numerically compared against three other FFNN1 training algorithms based on the impact on the implementation of the ADP control policy to a 4-dimensional hydropower reservoir problem, a 12-dimensional hydropower reservoir problem, and a 15-dimensional inventory problem. Results from both SAA and CAA showed that using FUA to approximate the value function yielded more accurate control solutions in less computation time.

2.A FFNN1 Training Error Distribution Plots

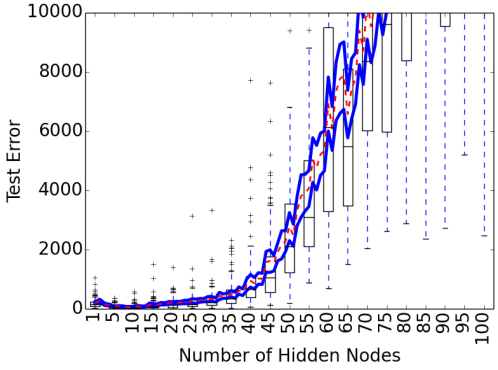
In the beginning of Section 2.4 it was described how each of the nine test problems $CP\text{-}L$ combined with either of the two error estimation methods yields a total of 18 hyperparamter test problems. Only four of these plots were shown in the text in Figure 2.1. All 18 are presented here.



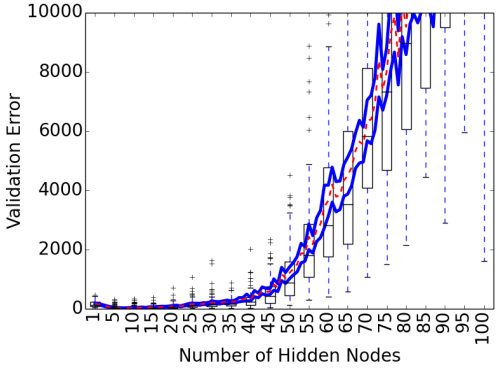
(a) test problem 4D-16, Component C2=TSM



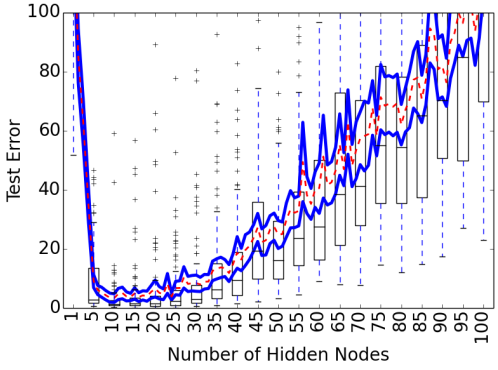
(b) test problem 4D-16, Component C2=VSM



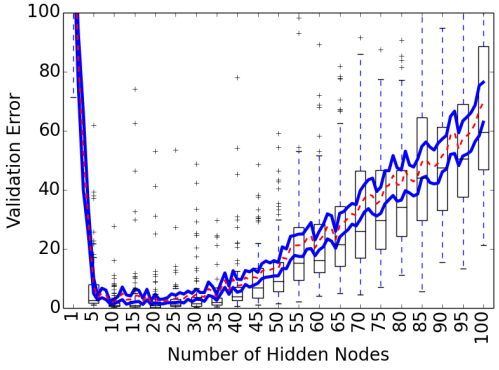
(c) test problem 4D-81, Component C2=TSM



(d) test problem 4D-81, Component C2=VSM

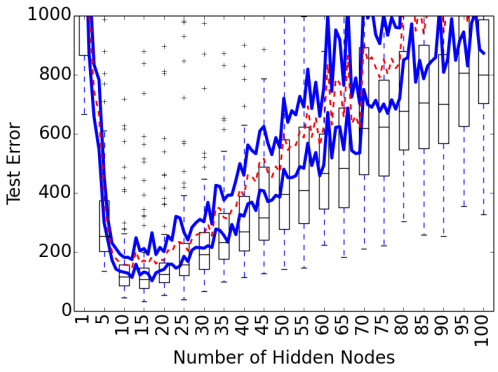


(e) test problem 4D-256, Component C2=TSM

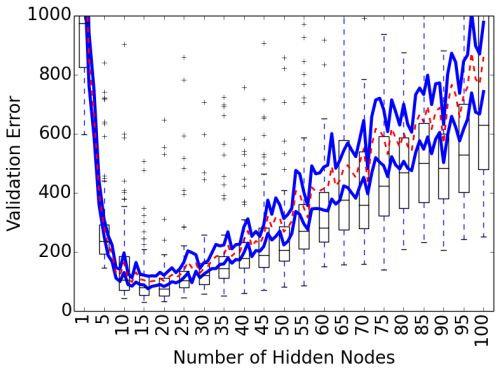


(f) test problem 4D-256, Component C2=VSM

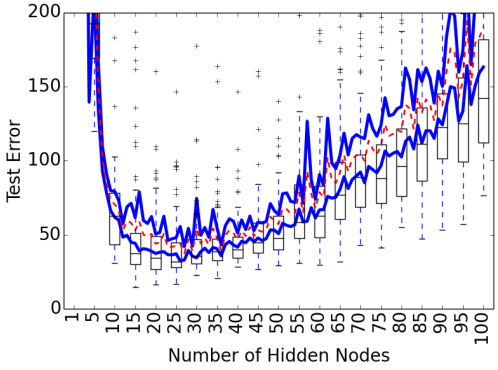
Figure 2.7: FFNN1 error distributions in 4D Hydropower problem. The solid lines show the 95% confidence interval of the estimated mean on the domain $\mathcal{H} = \{1, 2, ..., 100\}$, and the dashed line shows the mean. The boxplots show the distribution on domain $\mathcal{H} = \{1, 5, 10, ..., 100\}$.



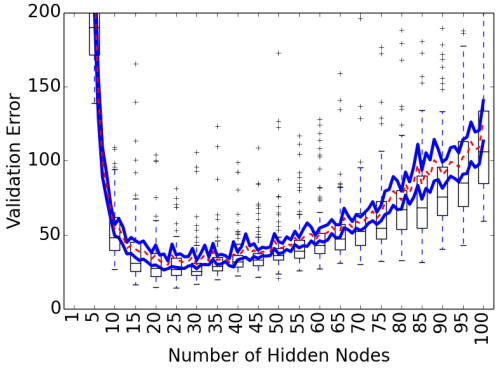
(a) test problem 12D-750, Component C2=TSM



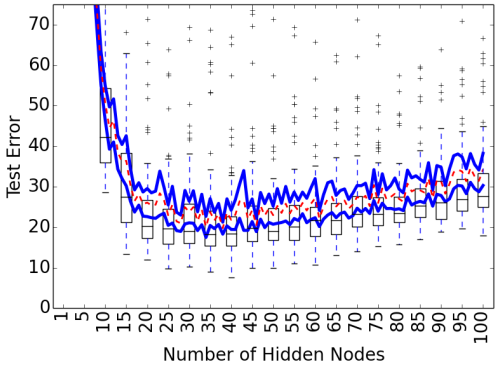
(b) test problem 12D-750, Component C2=VSM



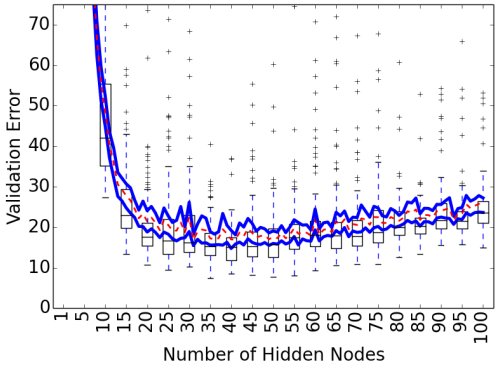
(c) test problem 12D-1500, Component C2=TSM



(d) test problem 12D-1500, Component C2=VSM

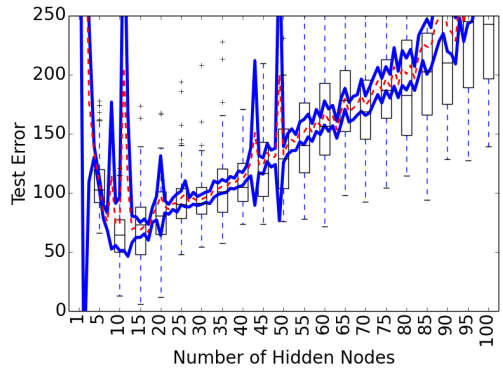


(e) test problem 12D-3000, Component C2=TSM

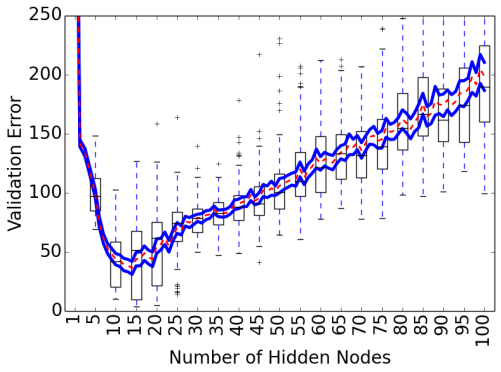


(f) test problem 12D-3000, Component C2=VSM

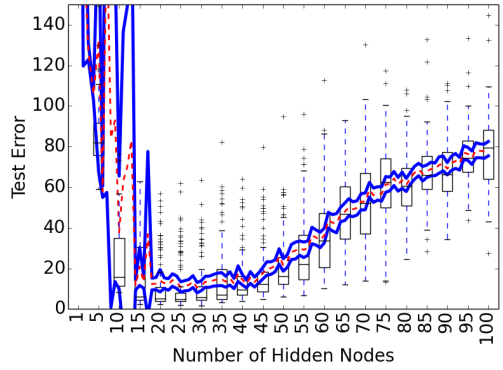
Figure 2.8: FFNN1 error distributions in 12D Hydropower problem. The solid lines show the 95% confidence interval of the estimated mean on the domain $\mathcal{H} = \{1, 2, ..., 100\}$, and the dashed line shows the mean. The boxplots show the distribution on domain $\mathcal{H} = \{1, 5, 10, ..., 100\}$.



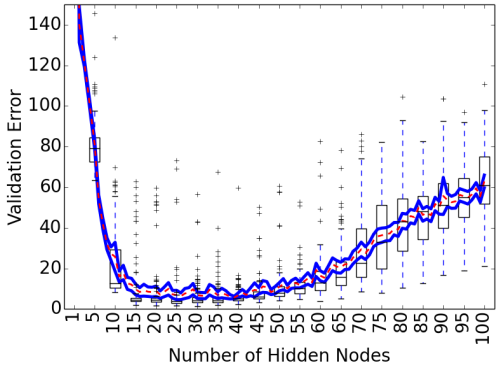
(a) test problem 15D-750, Component C2=TSM



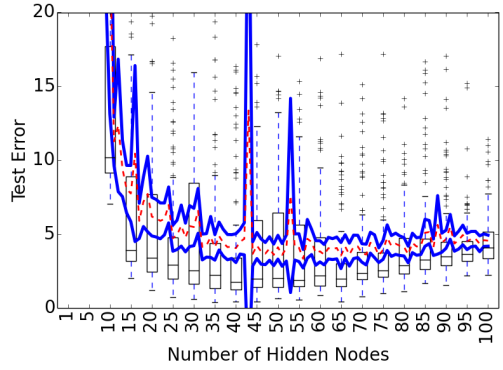
(b) test problem 15D-750, Component C2=VSM



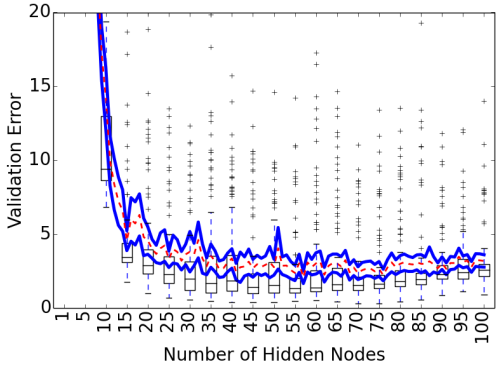
(c) test problem 15D-1500, Component C2=TSM



(d) test problem 15D-1500, Component C2=VSM



(e) test problem 15D-3000, Component C2=TSM



(f) test problem 15D-3000, Component C2=VSM

Figure 2.9: FFNN1 error distributions in 15D Inventory problem. The solid lines show the 95% confidence interval of the estimated mean on the domain $\mathcal{H} = \{1, 2, ..., 100\}$, and the dashed line shows the mean. The boxplots show the distribution on domain $\mathcal{H} = \{1, 5, 10, ..., 100\}$.

2.B Theoretical Properties of the Unimodal Approximation Optimization Algorithm

There are three results presented in Section 2.4.2 that require proofs.

2.B.1 Proof of Lemma 1

The first result is Lemma 1, and its proof makes use of Lemma 2.

Lemma 2. *Let $\mathcal{H}$ be a d-dimensional finite grid and select $H \in \mathcal{H}$ with the corresponding unimodal partial order $\preceq_H$. Suppose for subset $\mathcal{B} \subseteq \mathcal{H}$ a function $z : \mathcal{B} \to \mathbb{R}$ satisfies $z(i) \leq z(j)$ if $i \preceq_H j$ for $i, j \in \mathcal{B}$. Define $s = \min\{z(i) : i \in \mathcal{B}\}$ and define the function $z' : \mathcal{H} \to \mathbb{R}$ by*

$$z'(i) = \begin{cases} z(i) & \text{if } i \in \mathcal{B} \\ \max\{s\} \cup \{z(l) : l \in \mathcal{B}, l \preceq_H i\} & i \in \mathcal{H} \backslash \mathcal{B} \end{cases}$$

Then for all $i, j \in \mathcal{H}$ the relation $i \preceq_H j$ implies $z'(i) \leq z'(j)$, i.e. z' is a unimodal function on $\mathcal{H}$ with minimum H.

Proof. Select $i, j \in \mathcal{H}$. First, if $i, j \in \mathcal{B}$ and $i \preceq_H j$ then $z'(i) = z(i) \leq z(j) = z'(j)$. Second, if $i \in \mathcal{B}$, $j \notin \mathcal{B}$ and $i \preceq_H j$ then $z'(i) = z(i) \in \{z(l) : l \in \mathcal{B}, l \preceq_H j\} \Rightarrow z'(i) \leq z'(j)$. Third, suppose $i \in \mathcal{B}$, $j \notin \mathcal{B}$, and $j \preceq_H i$ but $z'(j) > z'(i)$. If $\{z(l) : l \in \mathcal{B}, l \preceq_H j\} \neq \emptyset$ then $\exists l$ such that $l \in \mathcal{B}$, $l \preceq_H j$, and $z(l) > z(i)$, but $l \preceq_H j \preceq_H i \Rightarrow l \preceq_H i$ which, because $i, l \in \mathcal{B}$, yields the contradiction that $z'(j) = z(l) \leq z(i) = z'(i)$; and, if $\{z(l) : l \in \mathcal{B}, l \preceq_H j\} = \emptyset$ then $z'(j) = s = \min\{z(l) : l \in \mathcal{B}\} \leq z(i) = z'(i)$ which contradicts the supposition. Finally, if $i, j \notin \mathcal{B}$ and $i \preceq_H j$ then $\{l \preceq_H i : l \in \mathcal{B}\} \subseteq \{l \preceq_H j : l \in \mathcal{B}\} \Rightarrow z'(i) \leq z'(j)$. □

The restatement and proof of Lemma 1 is now as follows:

Lemma 1. *Select k and $h, H \in \mathcal{H}$ such that at the k^{th} iteration of Algorithm 1 $n^k(h) = n^k(H) = 0$, and let $\preceq_h$ and $\preceq_H$ be the unimodal partial orders with minimum h and H (Definition 3). If $i \preceq_h j$ iff $i \preceq_H j$ for all $i, j \in \mathcal{H}$ such that $n^k(i) > 0$ and $n^k(j) > 0$, then $U(h, \mu^k, \sigma^k) = U(H, \mu^k, \sigma^k)$.*

Proof. Let z_1 be a unimodal function on $\mathcal{H}$ with minimum h such that $\sum_{i \in \mathcal{H}} \sigma^k(i)(z_1(i) - \hat{\mu}^k(i))^2 = U(h, \hat{\mu}^k, \sigma^k)$. Use Lemma 2 with $\mathcal{B} = \{i \in \mathcal{H} : n^k(i) > 0\}$, unimodal partial order $\preceq_H$, and function $z_1 : \mathcal{B} \to \mathbb{R}$ to obtain the unimodal function z' as defined in the lemma, so that z' is a unimodal function with minimum H that satisfies $z'(i) = z_1(i)$ for $i \in \mathcal{B}$. Therefore, since

$$\sum_{i \in \mathcal{H}} \sigma^k(i)(z_1(i) - \mu^k(i))^2 = \sum_{i \in \mathcal{B}} \sigma^k(i)(z_1(i) - \mu^k(i))^2 = \sum_{i \in \mathcal{B}} \sigma^k(i)(z'(i) - \mu^k(i))^2 = \sum_{i \in \mathcal{H}} \sigma^k(i)(z'(i) - \mu^k(i))^2$$

this shows that $U(h, \mu^k, \sigma^k) \geq U(H, \mu^k, \sigma^k)$. Since the reverse inequality is true by analogous reasoning this shows that $U(h, \mu^k, \sigma^k) = U(H, \mu^k, \sigma^k)$. □

2.B.2 Proof of Bounded θ

The second result stated in Section 2.4.2 is that the trained parameter values of θ are bounded when an FFNN1 is trained with the Levenberg-Marquardt algorithm. The formal statement and proof are as follows.

Lemma 3. *Suppose the Levenberg-Marquardt algorithm is used to train the FFNN1 using a sum of squared errors objective function on training data $\{(s_i, y_i)\}_{i=1}^L$, $s_i \in \mathbb{R}^{1Xn}, y_i \in \mathbb{R}$. The output of an FFNN1 F with h hidden nodes and parameters*

θ is $F(s_i;\theta,h) \in \mathbb{R}$. In the Levenberg-Marquard algorithm the parameter values θ^k at the k^{th} iteration are updated by

$$\theta^{k+1} = \theta^k - [J^T(\theta^k)J(\theta^k) + \mu^k I]^{-1} J^T(\theta^k) v(\theta^k), \tag{2.21}$$

where $v_i(\theta^k) = y_i - F(s_i;\theta^k,h)$; $J(\theta^k)_{i,j} = \frac{\partial v(\theta^k)_i}{\partial \theta_j^k}$ is the partial derivative of $v_i(\theta^k)$ with respect to the j^{th} parameter in θ^k, denoted θ_j^k; $\mu > 0$; and at the k^{th} iteration $\mu^k \geq \frac{\mu_0}{(\delta)^k}$ for some $\delta > 1$. Also, suppose F is twice differentiable.

If $\exists C^0 > 0$ such that $||\theta^0||_2 \leq C^0$ then for each k there exists some C^k such that $||\theta^k||_2 \leq C^k$.

Proof. Suppose $||\theta^k||_2 \leq C^k$. First, note that $J^T(\theta^k)J(\theta^k)$ is positive semidefinite with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$, and so $J^T(\theta^k)J(\theta^k) + \mu^k I$ is positive definite with smallest eigenvalue $\lambda_n + \mu^k > 0$. Then

$$||[J^T(\theta^k)J(\theta^k) + \mu^k I]^{-1}||_2 = \frac{1}{\lambda_n + \mu^k} \leq \frac{1}{\mu^k}.$$

Next, because F is twice differentiable the sum of squared errors $v^T(\theta)v(\theta)$ is also twice differentiable everywhere with respect to θ. This means there is some maximum gradient $2J^T(\theta)v(\theta)$ over the set of all $||\theta||_2 \leq C^k$. So, there exists some $\bar{C}^k$ such that $||J^T(\theta^k)v(\theta^k)||_2 \leq \bar{C}^k$. Therefore,

$$\begin{aligned} ||\theta^{k+1}||_2 &= ||\theta^k - [J^T(\theta^k)J(\theta^k) + \mu^k I]^{-1} J^T(\theta^k) v(\theta^k)||_2 \\ &\leq C^k + \bar{C}^k \frac{1}{\mu_k} \leq C^k + \bar{C}^k \frac{(\delta)^k}{\mu_0}. \end{aligned}$$

The result then follows from induction. □

The assumption that the initial parameter values θ^0 are bounded is true because in the MATLAB implementation the default Nguyen-Widrow initialization

method is used [27, 28]. Also, the MATLAB implementation Levenberg-Marquardt algorithm enforces a maximum number of allowed iterations.

2.B.3 Proof of Convergence Theorem

Theorem 1. *Set $I \geq 1$ and $\varepsilon > 0$ in the UAO algorithm (Algorithm 1). Select a d-dimensional finite grid $\mathcal{H}$, and for each $h \in \mathcal{H}$ let X_h be a random variable with mean $\mu(h) = \mathbb{E}(X_h)$ and compact support. Assume the function $\mu : \mathcal{H} \to \mathbb{R}$ is strictly unimodal with unique minimum h^*.*

Let $\mathcal{U}^1, \mathcal{U}^2, ...$ be the infinite sequence of random elements with probability measure P such that at the k^{th} iteration of the while loop (line 5 of Algorithm 1) $P(\mathcal{U}^k = \mathcal{U})$ is the probability that the solution in Equation (2.11) is $\mathcal{U} \in 2^{\mathcal{H}} \backslash \emptyset$ conditioned on the initial data in Line 3 of the UAO algorithm. Then

$$P(\lim_{k \to \infty} \mathcal{U}^k = \{h^*\}) = 1,$$

i.e. the UAO algorithm converges almost surely to the unique global minimum.

Proof. **Definitions**

The UAO is first slightly rewritten in order to rigorously express the functions μ^k, σ^k, n^k, $\mathbb{1}^k$, U^k, and $\mathcal{U}^k$ as random variables.

In each iteration of the *while* loop three random variables are sampled. In Line 7 an element $\tilde{h}$ is randomly chosen from a subset of $\mathcal{U}(\mu^k, \sigma^k)$, which can be simulated by randomly selecting an ordered permutation of $\mathcal{H}$ and selecting the first element in the permutation that is also in the given subset of $\mathcal{U}(\mu^k, \sigma^k)$. In Line 8 an element h is randomly chosen from the set $\{i \in \mathcal{H} : ||i - \tilde{h}||_1 \leq I\}$,

which can also be simulated using the same permutation trick. In Line 9 the random variable X_h is sampled. Let Y be the random element describing the probability of selecting two permutations from the set of permutations $Q(\mathcal{H})$ of $\mathcal{H}$ and obtaining samples of X_i for each $i \in \mathcal{H}$. Specifically, letting $(\Omega', \mathcal{F}', P')$ be a probability space then Y is a random element $Y : \Omega' \to Q(\mathcal{H}) \times Q(\mathcal{H}) \times \mathbb{R}^{|\mathcal{H}|}$. With $P_Q(\mathcal{A}) = |\mathcal{A}|/|Q(\mathcal{H})|$ for $\mathcal{A} \subseteq Q(\mathcal{H})$ the probability of selecting a set of permutations $\mathcal{A}$ and $P_i(r)$ the probability that $X_i < r$, the probability measure P' is defined as

$$P' = P_Q \times P_Q \prod_{i \in \mathcal{H}} \times P_i. \tag{2.22}$$

From Kolmogorov's Extension Theorem there is a unique probability space $(\Omega, \mathcal{F}, P)$ of the infinite sequence of iid random vectors $Y^1, Y^2, ...$

Given this sequence $Y^1, Y^2, ...$ the algorithm can be slightly rewritten. At iteration k of the *while* loop, replace lines 6 through 9 with

6: Get sample $y^i = (q_1, q_2, \tilde{x}^k(h_1), ..., \tilde{x}^k(h_{|\mathcal{A}|}))$ drawn from Y^k.

7: Order the set $\underset{h \in \mathcal{U}(\mu^k, \sigma^k)}{\operatorname{argmax}} \left\{ \min_{i \in \mathcal{H}_1^k} ||h - i||_1 \right\}$ according to q_1, set $\tilde{h}$ to first vertex

8: Order the set $\{i : i \in \mathcal{H}, ||\tilde{h} - i||_1 \leq I\}$ according to q_2, set v to first vertex

9: Set the sample $x^k(h)$ to $\tilde{x}^k(h)$.

Sequences of measurable functions f^k mapping from $Y^1, ..., Y^k$ to a measurable space are now defined. For $\omega \in \Omega$ each function $f^k(\omega)$ is calculated by setting $y^i = Y^i(\omega)$ in Line 6, iterating the *while* loop in the modified algorithm above $k - K$ times, and then setting f^k to the value of some variable that is defined in the UAO algorithm. Because all of the functions in the algorithm are measurable f^k is a sequence of random elements on probability space $(\Omega, \mathcal{F}, P)$. According to this procedure define μ_h^k, σ_h^k, n_h^k, $\mathbb{1}_h^k$, U_h^k, and $\mathcal{U}^k$ to be the values of $\mu^k(h)$, $\sigma^k(h)$,

$n^k(h)$, $\mathbb{1}^k(h)$, $U^k(h, \mu^k, \sigma^k)$, and $\mathcal{U}(\mu^k, \sigma^k)$, respectively.

There are three final definitions. Because it is assumed that X_h has compact support for each $h \in \mathcal{H}$, define B_h^L and B_h^U such that $P(X_h < B_h^L) = P(X_h > B_h^U) = 0$. Also, define

$$\Delta = \min\{\mu(j) - \mu(i) : i, j \in \mathcal{H}, i \preceq_{h^*} j\}, \tag{2.23}$$

and because it is assumed that μ is strictly unimodal $\Delta > 0$. Finally, select any $\delta \in (0, 1)$.

Objective

To prove that

$$P\big(\{\omega \in \Omega : \lim_{n\to\infty} U^k(\omega) = \{h^*\}\}\big) = 1 \tag{2.24}$$

we prove the equivalent statement that [29]

$$\lim_{m\to\infty} P(\{\omega \in \Omega : U^k(\omega) = \{h^*\} \forall n \geq m\}) = 1. \tag{2.25}$$

The proof shows that for a selected $\delta \in (0, 1)$ there exists some M and $\Omega' \in \mathcal{F}$ such that $P(\Omega') > \delta$ and that $U^k(\omega) = \{h^*\}$ for all $n \geq M$ for every $\omega \in \Omega'$.

Part I

By the Strong Law of Large Numbers there is some $\Omega_1 \in \mathcal{F}$ and M_1 such that 1) $P(\Omega_1) \geq \frac{\delta+1}{2}$; 2) for all $h \in \mathcal{H}$ and $\omega \in \Omega_1$ if $n_h^k(\omega) \geq M$ (i.e. h has been selected in Line 8 at least M times) then

$$|\mu_h^k(\omega) - \mu(h)| < \frac{1}{2}\Delta; \tag{2.26}$$

and 3) $B_h^L \leq \mu_h^k(\omega) \leq B_h^U$ for all $\omega \in \Omega_1$ and for all k. Define

$$C = \sum_{h \in \mathcal{H}} \max(1, \frac{M_1}{\varepsilon})(B_h^U - B_h^L)^2, \tag{2.27}$$

and we'll show $U_{h^*}^k(\omega) \leq C$ for all k and all $\omega \in \Omega_1$.

First, select $\omega \in \Omega_1$ and any k. Define $T = \{h \in \mathcal{H} : n_h^k(\omega) \geq M_1\}$. Also, observe that by definition $\sigma_v^k(\omega) \leq \max(1, \frac{n_h^k(\omega)}{\varepsilon})$ regardless of $n_h^k(\omega)$.

If $|T| = 0$, set $z(v) = \mu(v)$. Then z is a unimodal function with minimum h^* and

$$\sum_{h \in \mathcal{H}} \sigma_h^k(\omega)(z(h) - \mu_h^k(\omega))^2 = \sum_{h \in \mathcal{H}: n_h^k(\omega) > 0} \sigma_h^k(\omega)(z(h) - \mu_h^k(\omega))^2 \leq \tag{2.28}$$

$$\leq \sum_{h \in \mathcal{H}: n_h^k(\omega) > 0} \max(1, \frac{n_h^k(\omega)}{\varepsilon})(z(h) - \mu_h^k(\omega))^2 \leq C. \tag{2.29}$$

The last inequality is true because if $n_h^k(\omega) > 0$ then $B_h^L \leq \mu_h^k(\omega) \leq B_h^U \Rightarrow (z(h) - \mu_h^k(\omega))^2 \leq (B_h^U - B_h^L)^2$.

Next, if $|T| > 0$ and $h^* \in T$ use Lemma 2 with $\mathcal{B} = T$, $z(i) = \mu_i^k(\omega), i \in T$ and minimum h^* to obtain function z' that is unimodal with minimum h^* satisfying $z'(i) = \mu_i^k(\omega), i \in T$. The weighted sum of squares of z' is bounded by

$$\sum_{h \in \mathcal{H}} \sigma_h^k(\omega)(z'(h) - \mu_h^k(\omega))^2 = \sum_{h \in \mathcal{H} \setminus T: n_h^k(\omega) > 0} \sigma_h^k(\omega)(z'(h) - \mu_h^k(\omega))^2 \leq C. \tag{2.30}$$

Finally, if $|T| > 0$ but $h^* \notin T$, then use Lemma 2 with $\mathcal{B} = T \cup \{h^*\}$, $z(i) = \mu_i^k(\omega), i \in \mathcal{B}$ and minimum h^* to obtain a function z' that is unimodal function with minimum h^* satisfying $z'(i) = \mu_i^k(\omega), i \in T \cup \{h^*\}$, and again the weighted sum of squares is bounded by C.

Altogether, this shows that there exists a unimodal function z with minimum h^* such that $\sum_{h\in\mathcal{H}} \sigma_h^k(\omega)(z(h)-\mu_h^k(\omega))^2 \leq C$, i.e. $U_{h^*}^k(\omega) \leq C$.

Part II

Let $B = \max_{h\in\mathcal{H}} B_h^U - B_h^L$ and define

$$M_2 = C\frac{(4\varepsilon + B^2)^2}{2\varepsilon\Delta^2} + 1. \tag{2.31}$$

Also define the sets $\mathcal{H}_h = \{i \in \mathcal{H} : ||i-h||_1 \leq I\}$. Select any $\tilde{h} \in \mathcal{H}$. By the Strong Law of Large Numbers there is some $\Omega_{\tilde{h}}$ and $M_2(\tilde{h})$ such that 1) $P(\Omega_{\tilde{h}}) \geq \frac{\delta-1}{2|\mathcal{H}|} + 1$; and 2) for any $\omega \in \Omega_{\tilde{h}}$ if there are $M_2(\tilde{h})$ distinct iterations $k_1 < k_2 < \cdots < k_{M_2(\tilde{h})}$ such that $\tilde{h}$ is selected in Line 7 then

$$|\mu_i^k(\omega) - \mu(i)| < \frac{1}{4}\Delta \tag{2.32}$$

and

$$n_i^k(\omega) > M_2 \tag{2.33}$$

for each $i \in \mathcal{H}_{\tilde{h}}$ and for all $k \geq k_{M_2(\tilde{h})}$.

Select any $\omega \in \Omega_{\tilde{h}}$ and select $J \neq H \in \mathcal{H}_{\tilde{h}}$ such that $\mu(J) < \mu(H)$. Such a J and H exist because μ is strictly unimodal: because the domain $\mathcal{H}$ is a d-dimensional finite grid $\exists i \in \mathcal{H}_{\tilde{h}}$ such that either $i \preceq_{h^*} \tilde{h}$ or $\tilde{h} \preceq_{h^*} i$ which implies either $\mu(i) < \mu(\tilde{h})$ or $\mu(\tilde{h}) < \mu(i)$. Let $z : \mathcal{H} \to \mathbb{R}$ be any function such that $z(J) \geq z(H)$. If at iteration k, $\tilde{h}$ has been selected in Line 7 at least $M_2(\tilde{h})$ times

then

$$\sum_{h\in\mathcal{H}} \sigma_h^k(\omega)(z(h)-\mu_h^k(\omega))^2 \tag{2.34}$$

$$\geq \sum_{h\in\{J,H\}} \sigma_h^k(\omega)(z(h)-\mu_h^k(\omega))^2 \tag{2.35}$$

$$\geq \frac{\sigma_J^k(\omega)\sigma_H^k(\omega)}{\sigma_J^k(\omega)+\sigma_H^k(\omega)}(\mu_J^k(\omega)-\mu_H^k(\omega))^2 \tag{2.36}$$

The last line follows from solving

$$\begin{aligned} &\min_{z(J),z(H)} \sigma_J^j(\omega)(z(J)-\mu_J^k(\omega))^2+\sigma_H^k(\omega)(z(H)-\mu_H^k(\omega))^2 \\ &\text{s.t. } z(J)\geq z(H) \end{aligned} \tag{2.37}$$

with $\mu_J^k(\omega) < \mu_H^k(\omega)$ (because of the condition in Equation (2.32)), which has a solution at

$$z(J)=z(H)=\frac{\sigma_J^k(\omega)\mu_J^k(\omega)+\sigma_H^k(\omega)\mu_H^k(\omega)}{\sigma_J^k(\omega)+\sigma_H^k(\omega)}. \tag{2.38}$$

To continue this inequality chain, notice that the inequality $var(\{x_h^i : \mathbb{1}_h^i = 1, i=1,...,k\}) \leq \frac{B^2}{4}$ holds for $h=J,H$ because $B_h^L \leq x_h^i \leq B_h^U$ for $i=1,...,k$ if $\mathbb{1}_h^i = 1$. So by definition of σ_h^k there is a lower bound of

$$\sigma_h^k(\omega) \geq \frac{n_h^k(\omega)}{\varepsilon+\frac{B^2}{4}} \tag{2.39}$$

for $v=J,H$. Since $n_h^k(\omega) > M_2 > 1$ (from the condition in Equation (2.33)) for $v=J,H$, by definition of σ_h^k there is an upper bound $\sigma_h^k(\omega) \leq \frac{n_h^k(\omega)}{\varepsilon}$. Continuing the inequalities in Equation (2.36) the weighted sum of squared errors is then

bounded below by

$$\sum_{h\in\mathcal{H}} \sigma_h^k(\omega)(z(h) - x_h^k(\omega))^2 \tag{2.40}$$

$$\geq \frac{\sigma_J^k(\omega)\sigma_H^k(\omega)}{\sigma_J^k(\omega) + \sigma_H^k(\omega)}(\mu_J^k(\omega) - \mu_H^k(\omega))^2 \tag{2.41}$$

$$\geq \frac{\frac{n_J^k(\omega)n_H^k(\omega)}{(\varepsilon+B^2/4)^2}}{\frac{n_J^k(\omega)}{\varepsilon} + \frac{n_H^k(\omega)}{\varepsilon}}(\mu_J^k(\omega) - \mu_H^k(\omega))^2 \tag{2.42}$$

$$= \frac{\varepsilon}{(\varepsilon + B^2/4)^2}\frac{n_J^k(\omega)n_K^k(\omega)}{n_J^k(\omega) + n_K^k(\omega)}(\mu_J^k(\omega) - \mu_K^k(\omega))^2 \tag{2.43}$$

$$\geq \frac{\varepsilon}{(\varepsilon + B^2/4)^2}\frac{M_2M_2}{M_2 + M_2}(\mu_J^k(\omega) - \mu_K^k(\omega))^2 \tag{2.44}$$

$$\geq \frac{\varepsilon}{(\varepsilon + B^2/4)^2}\frac{M_2M_2}{M_2 + M_2}\Big(\frac{\Delta}{2}\Big)^2 \tag{2.45}$$

$$= M_2\frac{2\varepsilon\Delta^2}{(4\varepsilon + B^2)^2} > C \tag{2.46}$$

Altogether it has been shown that for all $\omega \in \Omega_2$ if 1) $J \neq H \in \mathcal{H}_{\tilde{h}}$ such that $\mu(J) < \mu(H)$; 2) $z : \mathcal{H} \to \mathbb{R}$ is any function such that $z(J) \geq z(H)$; and 3) $\tilde{h}$ has been selected in Line 7 at least $M_2(\tilde{h})$ times, then

$$\sum_{h\in\mathcal{H}} \sigma_h^k(\omega)(z(h) - \mu_h^k(\omega))^2 > C. \tag{2.47}$$

This implies two results.

First, suppose $\tilde{h} \neq h^*$. Because $\mathcal{H}$ is a d-dimensional finite grid there exists some $J \in \mathcal{H}_{\tilde{h}}$ such that $J \preceq_{h^*} \tilde{h}$. So, selecting $H = \tilde{h}$ it is true that $\mu(J) < \mu(H)$. Letting z be a unimodal function with minimum $\tilde{h}$ the inequality $z(J) \geq z(H)$ holds. This shows that if $\tilde{h} \neq h^*$ is selected at least $M_2(\tilde{h})$ times then $U_{\tilde{h}}^k(\omega) > C$.

Second, suppose $\tilde{h} = h^*$ and select any $i \in \mathcal{H}$ such that $i \neq h^*$. Let z be a unimodal function with minimum i. So, if z is a unimodal function with minimum i then $z(i) \leq z(h^*)$, but, because μ is strictly unimodal the strict inequality $\mu(i) >$

$\mu(h^*)$ holds. This shows that if $\tilde{h} = h^*$ is selected at least $M_2(h^*)$ times then $U_i^k(\omega) > C$ for all $i \in \mathcal{H}$ with $i \neq h^*$.

Part III

Parts I and II are now combined to prove the result. In general, for any $\Omega_i \in \mathcal{F}, i = 1, ..., m$ the probability of their intersection is bounded below by

$$P(\cap_{i=1,...,m}\Omega_i) \geq \sum_{i=1}^{m} P(\Omega_i) - (m-1). \tag{2.48}$$

Let the intersection of the sets defined in the previous two Parts be $\bar{\Omega} = \Omega_1 \cap_{h\in\mathcal{H}} \Omega_h$. The probability of $\bar{\Omega}$ is therefore at least

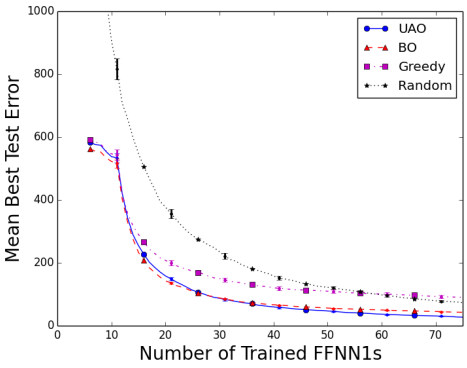
$$P(\Omega_1 \cap_{h\in\mathcal{H}} \Omega_h) \geq \frac{\delta+1}{2} + |\mathcal{H}|(\frac{\delta-1}{2|\mathcal{H}|} + 1) + (|\mathcal{H}| + 1 - 1) = \delta. \tag{2.49}$$

Set $M = |\mathcal{H}| * \max_{h\in\mathcal{H}} M_2(h)$ and select any $\omega \in \bar{\Omega}$. From Part I $U_{h^*}^k(\omega) \leq C$ for all k. From Part II if at iteration k any $h \in \mathcal{H}\backslash\{h^*\}$ is selected in line 7 at least $M_2(h)$ times then $U_h^k(\omega) > C$, and so h will not be selected in Line 7 again. By the pigeon hole principle, h^* must therefore be selected in Line 7 at least $M_2(h^*)$ times by the M^{th} iteration. Finally, combining this with the second result in Part II shows that for $k > M$ and for all $\omega \in \bar{\Omega}$ that 1) $U_i^k(\omega) > C$ for all $i \in \mathcal{H}\backslash\{h^*\}$ and 2) $U_{h^*}^k(\omega) \leq C$. That is, $\mathcal{U}^k = \{h^*\}$ for all $k > M$.

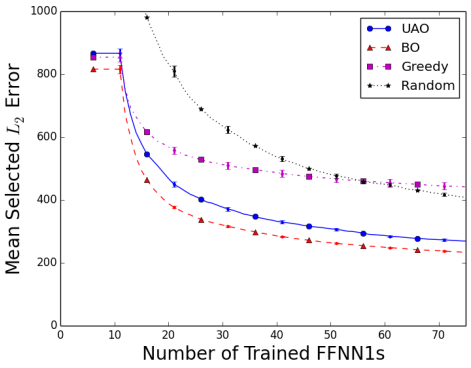
□

2.C Progress Plots

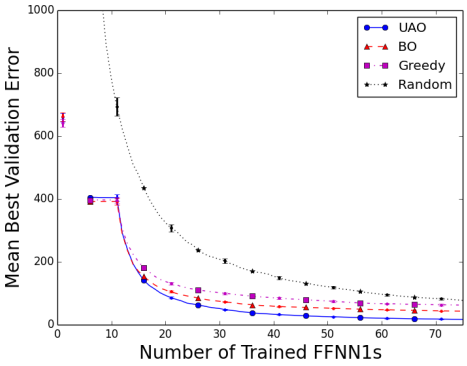
All of the progress plots described in Section 2.4.3 are shown here.



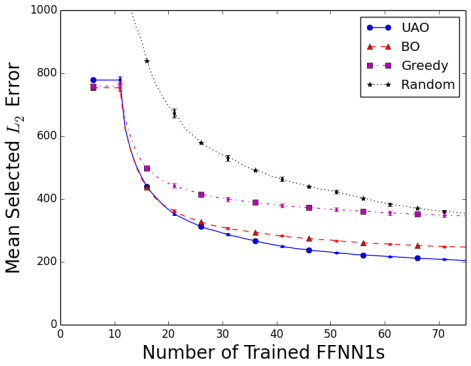
(a) best test error



(b) L_2 error of FFNN1 with best test error

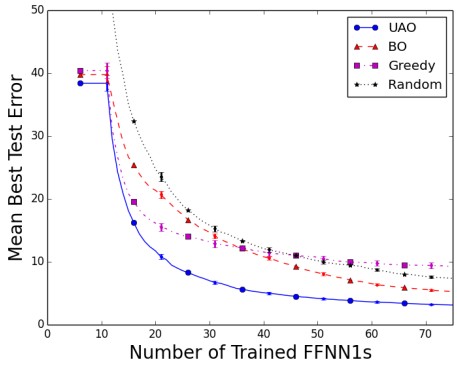


(c) best validation error

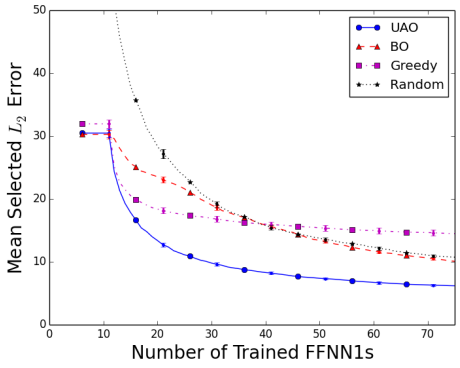


(d) L_2 error of FFNN1 with best validation error

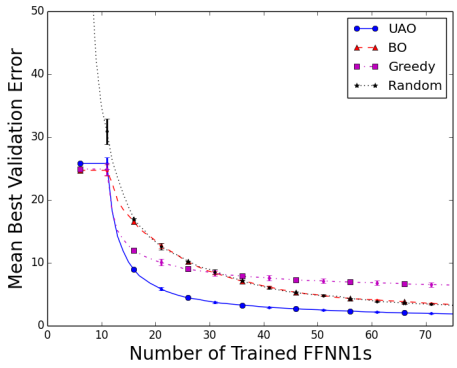
Figure 2.10: Progress plots comparing optimization algorithms applied to test problem 4D-16.



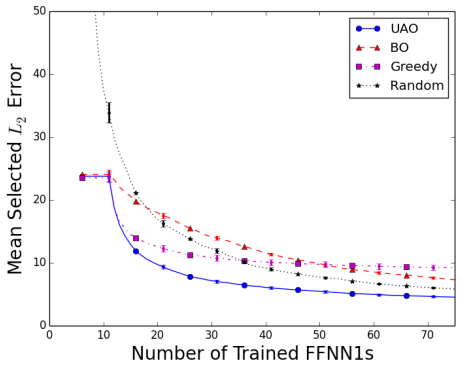
(a) best test error



(b) L_2 error of FFNN1 with best test error

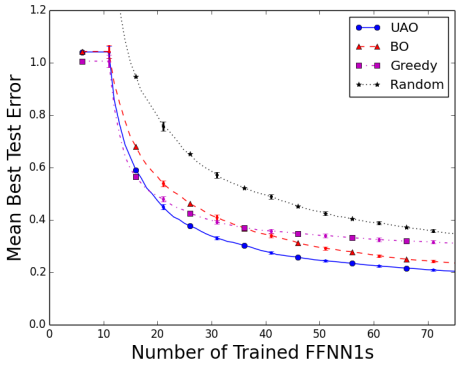


(c) best validation error

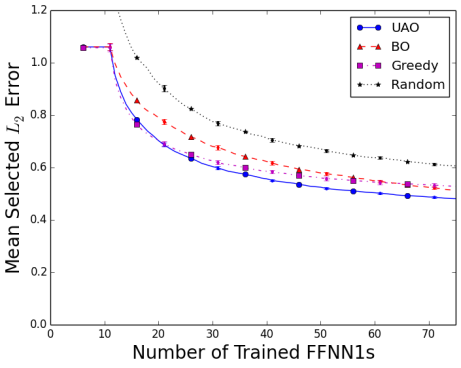


(d) L_2 error of FFNN1 with best validation error

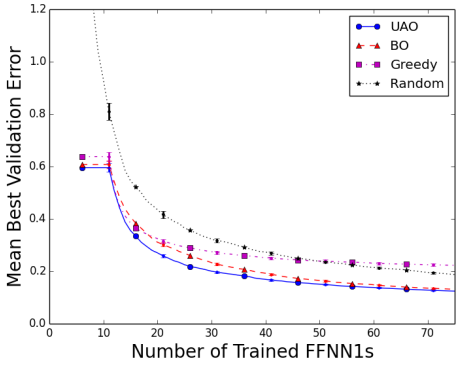
Figure 2.11: Progress plots comparing optimization algorithms applied to test problem 4D-81.



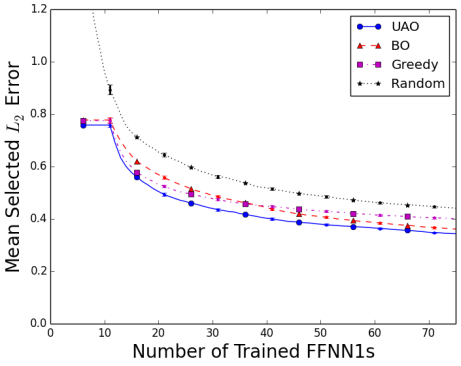
(a) best test error



(b) L_2 error of FFNN1 with best test error

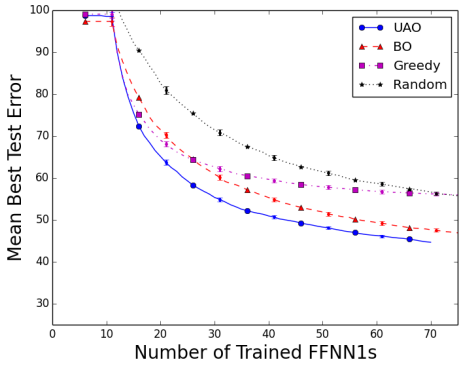


(c) best validation error

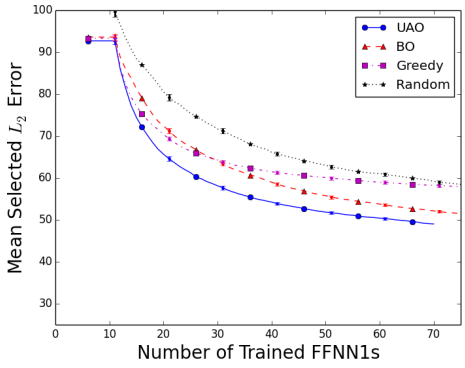


(d) L_2 error of FFNN1 with best validation error

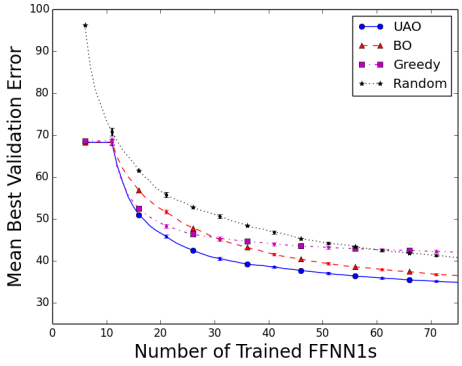
Figure 2.12: Progress plots comparing optimization algorithms applied to test problem 4D-256.



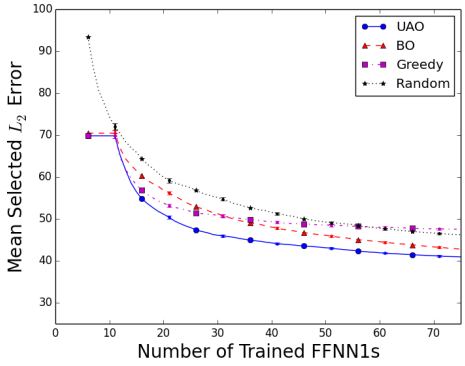
(a) best test error



(b) L_2 error of FFNN1 with best test error

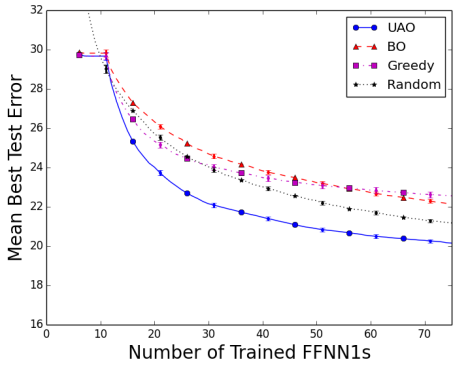


(c) best validation error

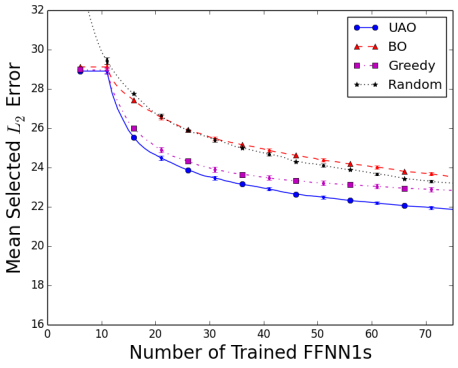


(d) L_2 error of FFNN1 with best validation error

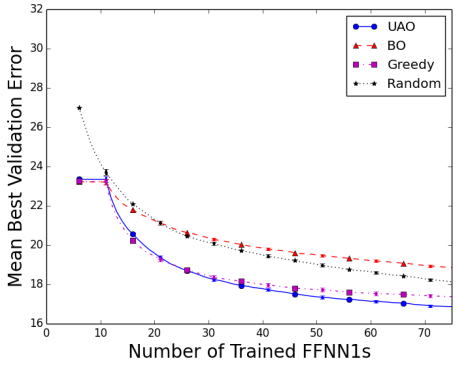
Figure 2.13: Progress plots comparing optimization algorithms applied to test problem 12D-750.



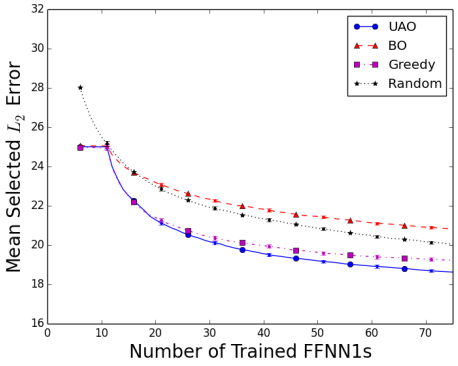
(a) best test error



(b) L_2 error of FFNN1 with best test error

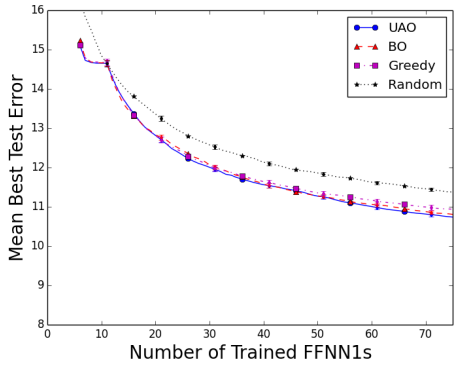


(c) best validation error

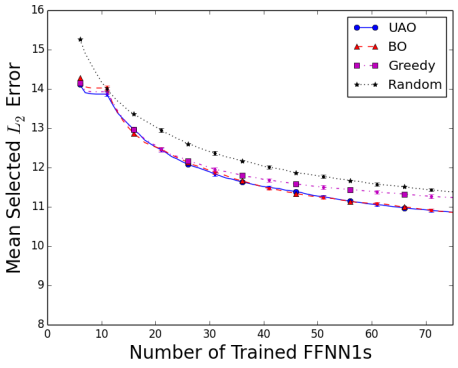


(d) L_2 error of FFNN1 with best validation error

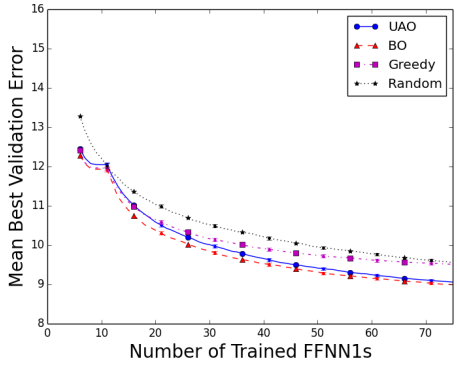
Figure 2.14: Progress plots comparing optimization algorithms applied to test problem 12D-1500.



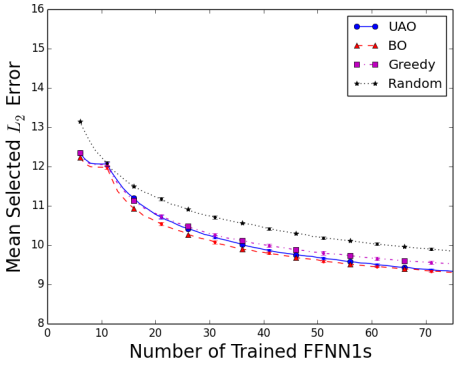
(a) best test error



(b) L_2 error of FFNN1 with best test error

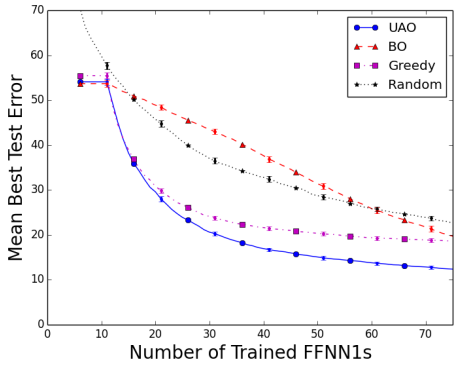


(c) best validation error

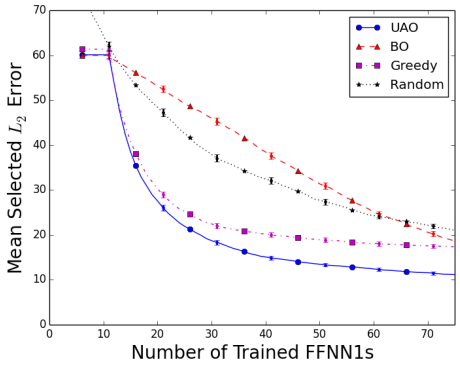


(d) L_2 error of FFNN1 with best validation error

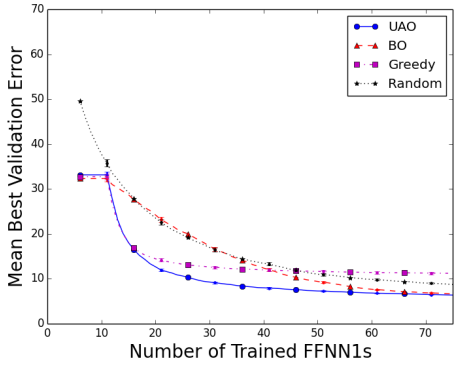
Figure 2.15: Progress plots comparing optimization algorithms applied to test problem 12D-3000.



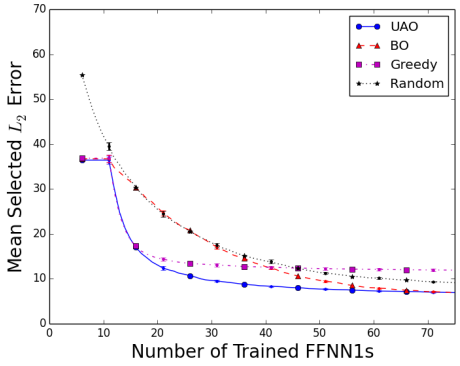
(a) best test error



(b) L_2 error of FFNN1 with best test error

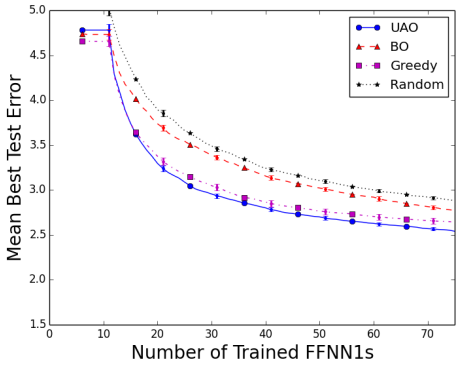


(c) best validation error

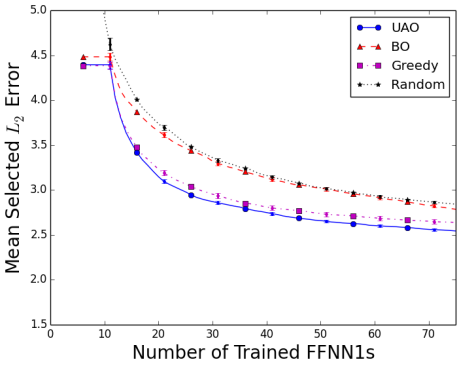


(d) L_2 error of FFNN1 with best validation error

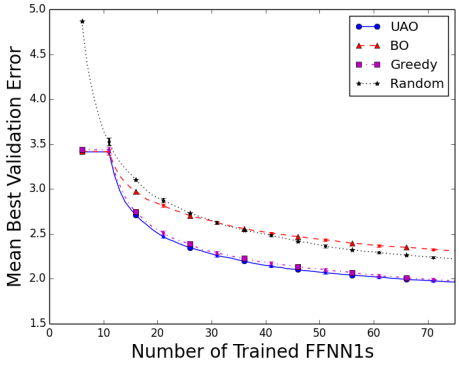
Figure 2.16: Progress plots comparing optimization algorithms applied to test problem 15D-750.



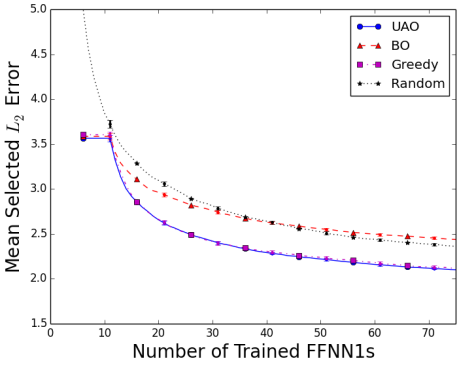
(a) best test error



(b) L_2 error of FFNN1 with best test error

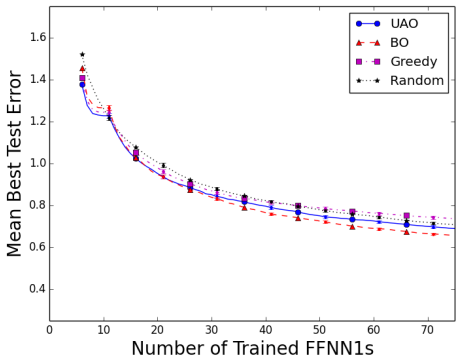


(c) best validation error

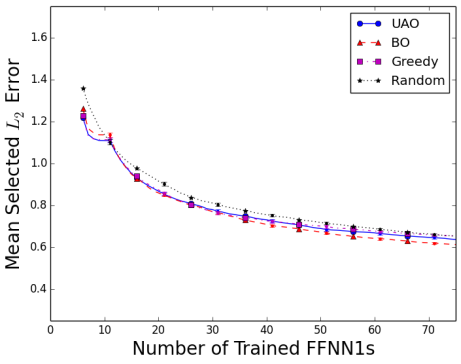


(d) L_2 error of FFNN1 with best validation error

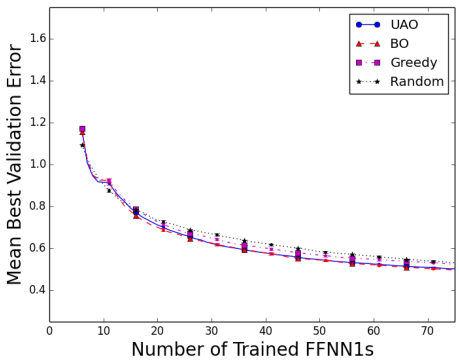
Figure 2.17: Progress plots comparing optimization algorithms applied to test problem 15D-1500.



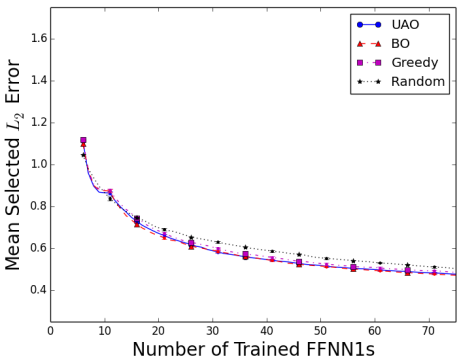
(a) best test error



(b) L_2 error of FFNN1 with best test error



(c) best validation error



(d) L_2 error of FFNN1 with best validation error

Figure 2.18: Progress plots comparing optimization algorithms applied to test problem 15D-3000.

2.D Speedup Definition

Suppose an optimization algorithm ALG is applied to a minimization problem multiple times. After t seconds of computation time let the mean performance of the algorithm, with respect to negatively oriented metric *error* (i.e. more negative is better), over the multiple trials be $\mu^{error}_{ALG}(t)$. The performance of algorithm ALG1 can be compared to the performance of algorithm ALG2 by using a t-test at the 5% confidence level to compute

$$worst^{error}_{ALG1,ALG2}(t) = \min\{t' \geq 0 : \text{fail to reject hypothesis that } \mu^{error}_{ALG1}(t') \leq \mu^{error}_{ALG2}(t)\}. \tag{2.50}$$

This represents the worst-case scenario in terms of how well ALG2 performs compared to ALG1. If $worst^{error}_{ALG1,ALG2}(t) > t$ then there is sufficient evidence to state that ALG2 yields a more negative (better) performance after t seconds than ALG1, i.e. ALG2 outperformed ALG1. Conversely, define

$$best^{error}_{ALG1,ALG2}(t) = \min\{t' \geq 0 : \text{reject hypothesis that } \mu^{error}_{ALG1}(t') \geq \mu^{error}_{ALG2}(t)\}, \tag{2.51}$$

which represents the best case for ALG2. If $best^{error}_{ALG1,ALG2} < t$ then there is sufficient evidence to state that ALG1 yields a more negative (better) performance after t seconds than ALG2, i.e. ALG1 outperformed ALG2.

Notice that $worst^{error}_{ALG1,ALG2}(t) \leq best^{error}_{ALG1,ALG2}(t)$. The speedup $S^{error}_{ALG1,ALG2}$ can then be defined as

$$S^{error}_{ALG1,ALG2}(t) = \begin{cases} worst^{error}_{ALG1,ALG2}(t)/t & \text{if } worst^{error}_{ALG1,ALG2}(t) > t \\ & \text{or } best^{error}_{ALG1,ALG2}(t) < t \\ 1 & \text{else} \end{cases} \tag{2.52}$$

In the second line there is neither sufficient evidence to state that ALG2 is better or worse. From the definition if $S^{error}_{ALG1,ALG2}(t) > 1$ then ALG2 is *at least* $S^{error}_{ALG1,ALG2}(t)$ times faster than ALG1 as measured at time t; if $S^{error}_{ALG1,ALG2}(t) < 1$ then ALG2 is *no more than* $S^{error}_{ALG1,ALG2}(t)$ times slower than ALG1.

2.E Computation and Accuracy Analysis Trials

The plots below show the Computation and Accuracy Analysis results. These are the same as in Figure 2.5, except that here the results are also shown for the individual trials. Markers are denoted by shape, color, and whether they are filled or unfilled. Markers of the same color use the same FFNN1 fitting and markers of the same shape use the same number of state space samples.

Individual trials are shown with unfilled markers. The y-axis value is the MEC $\bar{C}'$ in Equation (2.19). The error bars are given by $\sigma^2(\bar{C}')$ in Equation (2.20), although they are often smaller than the marker size and cannot be seen. The means over the five trials are shown with the filled markers.

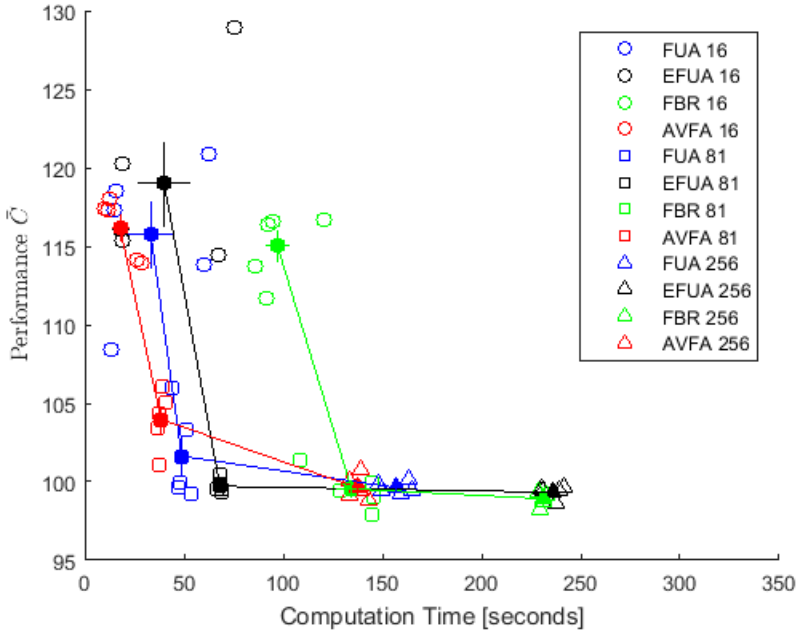


Figure 2.19: Computation and Accuracy Analysis for the 4D Hydropower test control problem.

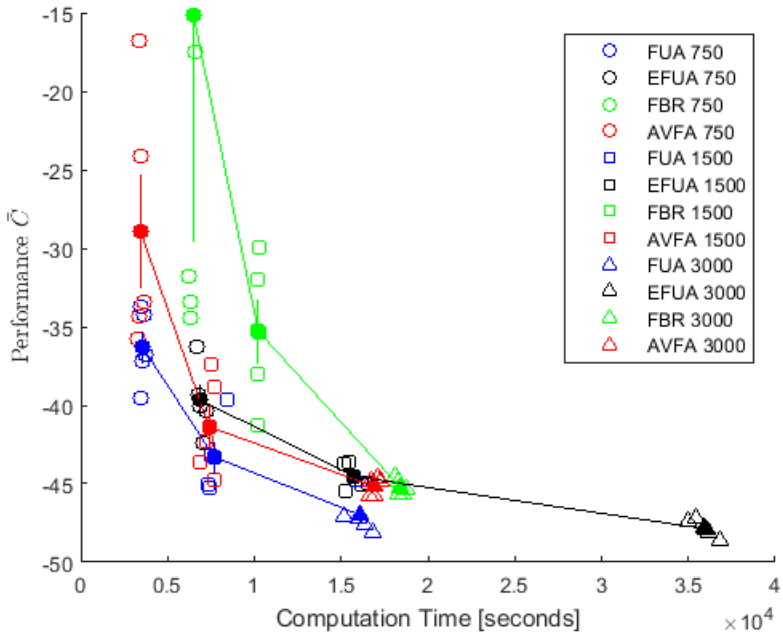


Figure 2.20: Computation and Accuracy Analysis for the 12D Hydropower test control problem. Some BR data points have MEC values too large (bad) to fit on the plot.

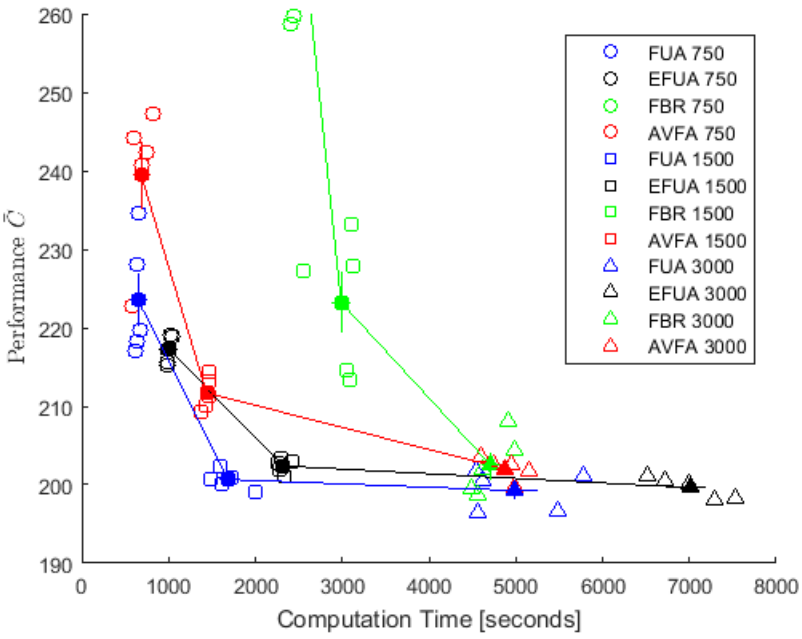


Figure 2.21: Computation and Accuracy Analysis for the 15D Inventory test control problem. Some BR data points have MEC values too large (bad) to fit on the plot.

2.F Computation Time Plots

In Section 2.6 the computation time breakdown was only shown for the 12D Hydropower reservoir. The time results for all three test control problems are shown here.

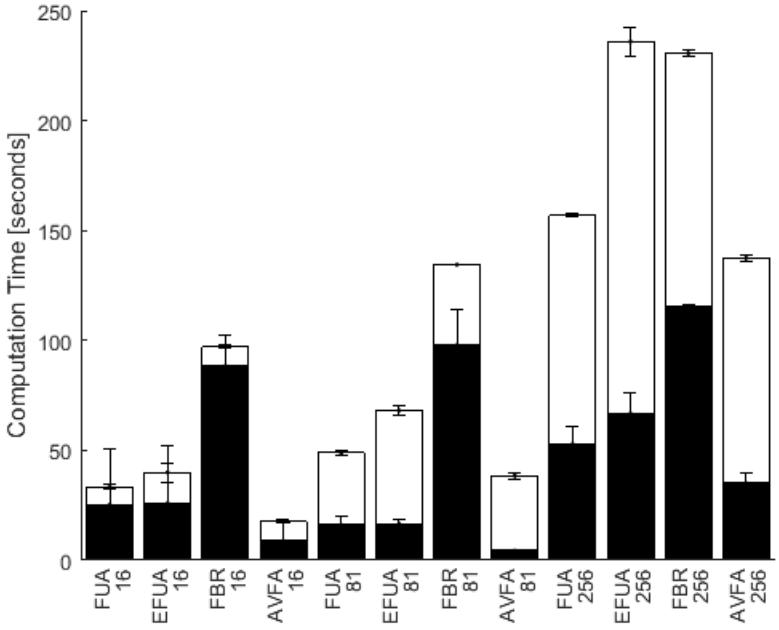


Figure 2.22: Computation time of the 4D Hydropower problem. Black bars are training time, white bars are time to evaluate state space samples, and bar height is total time.

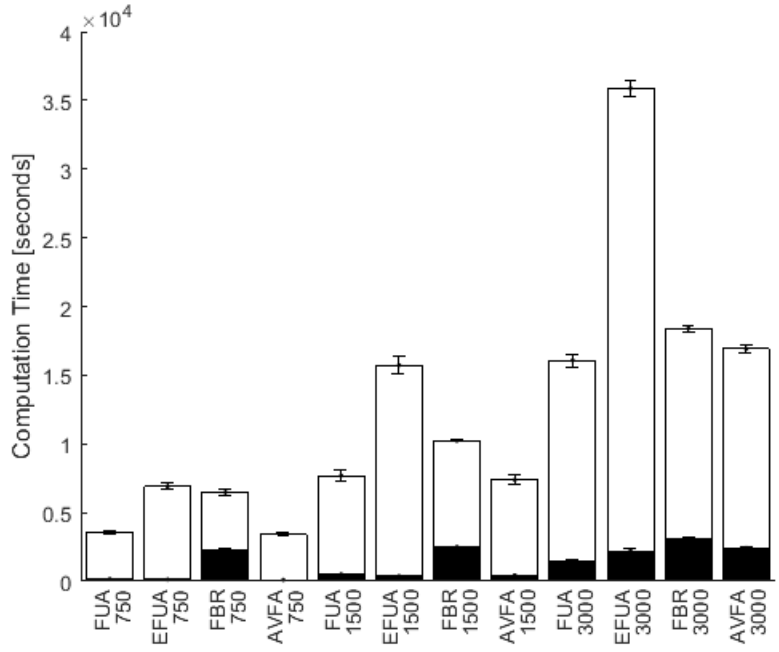


Figure 2.23: Computation time of the 12D Hydropower problem. Black bars are training time, white bars are time to evaluate state space samples, and bar height is total time.

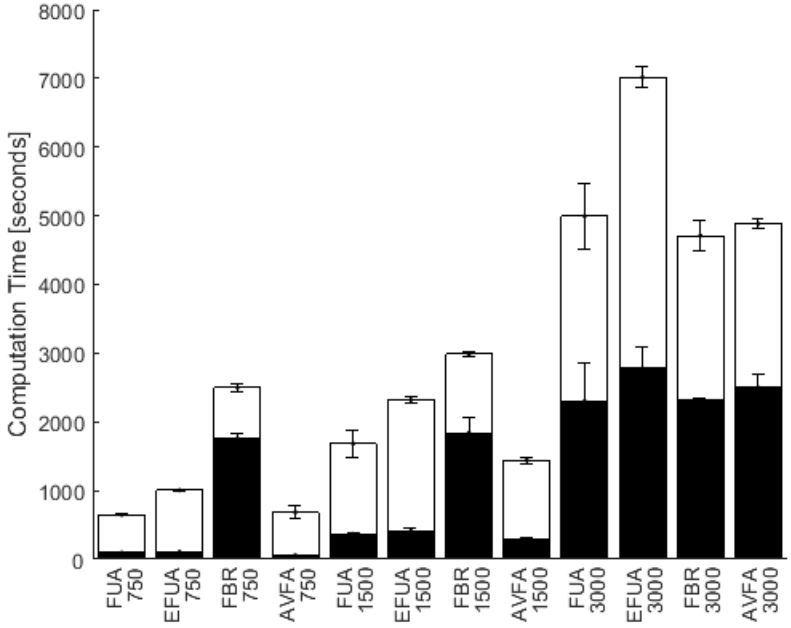


Figure 2.24: Computation time of the 15D Inventory problem. Black bars are training time, white bars are time to evaluate state space samples, and bar height is total time.

BIBLIOGRAPHY

[1] Andrew R Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1):115–133, 1994.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[3] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[4] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] Andrea Castelletti, Daniele de Rigo, Andrea Emilio Rizzoli, Rodolfo Soncini-Sessa, and Enrico Weber. Neuro-dynamic programming for designing water reservoir network management policies. *Control Engineering Practice*, 15(8):1031–1038, 2007.

[7] Cristiano Cervellera, Victoria CP Chen, and Aihong Wen. Optimization of a large-scale water reservoir network by stochastic dynamic programming with efficient state space discretization. *European Journal of Operational Research*, 171(3):1139–1151, 2006.

[8] Cristiano Cervellera, Mauro Gaggero, and Danilo Macciò. Low-discrepancy sampling for approximate dynamic programming with local approximators. *Computers & Operations Research*, 43:108–115, 2014.

[9] Cristiano Cervellera and Marco Muselli. Efficient sampling in approximate dynamic programming algorithms. *Computational Optimization and Applications*, 38(3):417–443, 2007.

[10] Cristiano Cervellera, Aihong Wen, and Victoria CP Chen. Neural network and regression spline value function approximations for stochastic dynamic programming. *Computers & operations research*, 34(1):70–90, 2007.

[11] Victoria CP Chen. Application of orthogonal arrays and mars to inventory forecasting stochastic dynamic programs. *Computational statistics & data analysis*, 30(3):317–341, 1999.

[12] Victoria CP Chen. Measuring the goodness of orthogonal array discretizations for stochastic programming and stochastic dynamic programming. *SIAM Journal on Optimization*, 12(2):322–344, 2002.

[13] Victoria CP Chen, David Ruppert, and Christine A Shoemaker. Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming. *Operations Research*, 47(1):38–53, 1999.

[14] Richard Combes and Alexandre Proutiere. Unimodal bandits: Regret lower bounds and optimal algorithms. In *ICML*, pages 521–529, 2014.

[15] Chelsia Amy Doukim, Jamal Ahmed Dargham, and Ali Chekima. Finding the number of hidden neurons for an mlp neural network using coarse to fine search technique. In *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*, pages 606–609. IEEE, 2010.

[16] Huiyuan Fan, Prashant K Tarun, and Victoria CP Chen. Adaptive value function approximation for continuous-state stochastic dynamic programming. *Computers & Operations Research*, 40(4):1076–1084, 2013.

[17] F Dan Foresee and Martin T Hagan. Gauss-newton approximation to bayesian learning. In *Neural Networks, 1997., International Conference on*, volume 3, pages 1930–1935. IEEE, 1997.

[18] Alexander IJ Forrester and Andy J Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79, 2009.

[19] Efi Foufoula-Georgiou and Peter K Kitanidis. Gradient dynamic programming for stochastic optimal control of multidimensional water resources systems. *Water resources research*, 24(8):1345–1359, 1988.

[20] Janis Hardwick and Quentin F Stout. Optimizing a unimodal response function for binary variables. In *Optimum Design 2000*, pages 195–210. Springer, 2001.

[21] Y Yu Jia and Shie Mannor. Unimodal bandits. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 41–48, 2011.

[22] Sharon A Johnson, Jery R Stedinger, Christine A Shoemaker, Ying Li, and Jose Alberto Tejada-Guibert. Numerical solution of continuous-state dynamic programs using linear and spline interpolation. *Operations Research*, 41(3):484–500, 1993.

[23] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238, 1995.

[26] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[27] MathWorks. initnw. `http://www.mathworks.com/help/nnet/ref/initnw.html`, 2015. Online; accessed October 13, 2015.

[28] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 21–26. IEEE, 1990.

[29] Hossein Pishro-Nik. *Introduction to Probability, STatistics, and Random Processes*. Kappa Research, 2014.

[30] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[31] Rommel G Regis and Christine A Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global optimization*, 31(1):153–171, 2005.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[33] Kazuhiro Shin-Ike. A two phase method for determining the number of neurons in the hidden layer of a 3-layer neural network. In *SICE Annual Conference 2010, Proceedings of*, pages 238–242. IEEE, 2010.

[34] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[35] Quentin F Stout. Optimal algorithms for unimodal regression. *Ann Arbor*, 1001:48109–2122, 2000.

[36] Shuxiang Xu and Ling Chen. A novel approach for determining the optimal number of hidden layer neurons for fnns and its application in data mining. 2008.

[37] Sidney Yakowitz. Dynamic programming applications in water resources. *Water resources research*, 18(4):673–696, 1982.

[38] William W-G Yeh. Reservoir management and operations models: A state-of-the-art review. *Water resources research*, 21(12):1797–1818, 1985.

CHAPTER 3

GENERATING LONG-TERM WIND SCENARIOS CONDITIONED ON SEQUENTIAL SHORT-TERM FORECASTS

3.1 Abstract

The control of power systems with wind integration is often conditioned on the most recent wind power forecasts, and their performance depends upon the wind power outcome. These power systems can therefore be evaluated by simulating them on a sequence of wind power forecasts and outcomes and then calculating a performance metric. This Chapter develops the Long Term Generation (LTG) method that generates synthetic long-term wind power outcome scenarios conditioned on sequential short-term historical forecasts. These synthetic scenarios can be used in the simulation process, and they can therefore help to better evaluate a power system. Because the joint distribution of wind forecasts and wind outcomes impacts power system performance, the Joint Distribution Comparison (JDC) test is also developed to test the hypothesis that the joint distribution of historical forecasts and synthetic scenarios is the same as the joint distribution of historical forecasts and historical outcomes. The new LTG method is applied to a dataset of historical wind forecasts and outcomes from Bonneville Power Administration, and the synthetic scenarios are evaluated with the JDC test.

3.2 Nomenclature

Δ wind forecasts extend Δ time steps

$\omega^t(j)$ for $j = 1, 2, ...,$ synthetic wind scenario j of wind power occurring at time step t

F_i^t for $i \in \{1, ..., \Delta\}$, random variable; point forecast of wind power that will occur at time step $t + i$ from forecast that was generated at time t

f_i^t for $i \in \{0, ..., \Delta\}$, historical sample from F^t

F^t multivariate random vector of forecasts $F^t = (F_1^t, ..., F_\Delta^t)$ generated at time step t

f^t historical forecasts vector, $f^t = (f_1^t, ..., f_\Delta^t)$

P probability measure, $P(W^{T_1}, F^{T_1}, ...W^{T_2}, F^{T_2})$

$\hat{P}$ estimated approximation of P

W^t random variable of wind outcome that occurs at time step t

w^t historical wind outcome that occurred at time step t; sample from W^t

T_1, T_2 long-term scenarios are generated from time step T_1 to T_2

T_1', T_2' with $T_1' \leq T_1$ and $T_2' \geq T_2$; use historical wind data from T_1' to T_2' to generate long-term scenarios over time steps T_1 to T_2

$X^{t,u}$ random vector that is a function of wind forecasts; $X^{t,u}$ are predictor variables of W^{t+u}. e.g. with $X^{t,u} = F_1^{t+u-1}$, we can estimate $P(W_0^t | F_1^{t-1})$ with data $\{(F_1^{t+u-1}, W_0^{t+u})\}_u$.

$x^{t,u}$ historical value of $X^{t,u}$

3.3 Introduction

As wind and solar energy production has increased many techniques have been developed to help determine the best ways to make use of these renewable energy sources and integrate them into the power grid. For example, this includes unit commitment problems where wind or solar power can be balanced with thermal generators [34, 35], geothermal generators [19], pumped hydro storage [20], a combination of controllable energy sources [33, 32], and batteries [18]. In this Chapter

we develop techniques that can be used with wind or solar energy, but in the remainder of the Chapter we will use a wind data set.

A crucial step in the development of these renewable energy power systems is evaluating how well they perform. Evaluation can be performed through simulation, where a sequence of wind power forecasts and wind power outcomes is first selected, the power system is simulated using this wind data sequence, and a resulting performance metric is calculated, such as total profit. This evaluation should use multiple wind data sequences in order to determine the distribution of performances over possible wind outcomes. Additionally, long-term wind sequences should be used in order to evaluate the long-term performance, because short-term performance estimates may be biased by initial conditions or may not capture system dynamics that only appear over a long horizon.

Literature Review

There are three approaches for obtaining wind forecast and outcome time series to use for power system simulation. The first approach is to use historical forecasts and historical outcomes, which provides a single long-term wind data sequence. This approach was used in [4] and [3], where the power systems were simulated over two months and one year, respectively.

A second approach is to obtain a long-term sequence of wind outcomes and then generate synthetic wind forecasts at each time step that are conditioned on the outcomes. In [33] the WILMAR Scenario Tree Tool (STT) [5] was used to generate forecast scenarios at every hour conditioned on a time series of hourly wind data spanning one year, and a power system was simulated on this data. STT generates these forecasts by using an ARMA model to generate a sequence

of forecast errors. A power system was simulated over 91 days in [35] using a wind data sequence that was generated using a similar process, as detailed in [13]. This approach assumes the wind forecasts errors of forecasts generated at different time steps are mutually independent, which may not be correct.

The third approach is to use a sequence of historical forecasts and then generate synthetic wind outcomes conditioned on the forecasts. All existing methods following this approach only generate short-term synthetic wind scenarios conditioned on a single short-term wind forecast. Specifically, these methods generate wind scenarios over time steps $t+1, ..., t+\Delta$ conditioned on the single wind forecast that was generated at time t and which predicts wind power at time steps $t+1, ..., t+\Delta$. These short-term scenario generation methods, developed in [28, 22, 23], all follow the same two-step approach of 1) estimating the marginal distribution of wind power at each time step $t+k$ for $k = 1, ..., \Delta$ conditioned on the point forecast information, and then 2) estimating the full joint distribution from which scenarios can be drawn. This approach will be referred to as the marginal-then-joint algorithm.

A diverse set of techniques have been developed that can be used to accomplish the first step of the marginal-then-joint algorithm, which is estimating the marginal distributions of wind power at a single time step conditioned on a point forecast. Continuous parametric distributions have been examined such as the Beta distribution [8], as well as mixed discrete-continuous distributions that combined delta distributions with Laplace [31] and generalized logit-Normal [25] distributions. Quantile regression has been applied [10, 28] and compared to other techniques like a local Gaussian model and a Nadaraya-Watson estimator [11]. A time adaptive quantile regression model, which assumes the quantiles are slowly

time varying, was found to improve results [24]. Similar to quantile regression are interval estimators, where [27] partitions forecast conditions into different cases, estimates probability intervals for each case, and uses fuzzy logic to combine the intervals together. Empirical distributions were created by binning the forecasts and sorting the historical data points according to bin [22]. A last class of methods is kernel density estimation [6, 21], and a more recent extension created a time-adaptive kernel density estimation using the Nadaraya-Watson model [7].

In the second step of the marginal-then-joint algorithm, which is constructing the joint distribution over multiple time steps of the individual marginal distributions, a Gaussian copula approach is used in all of [28, 22, 23]. In this approach the marginal distributions are transformed into standard normal distributions, and the joint distribution is uniquely specified by the multivariate normal covariance matrix. In [28] a time-adaptive exponential-forgetting scheme is used that updates the covariance matrix as more data arrives. Another approach uses an exponential covariance function that is defined by a distance parameter [26], and a method for estimating this parameter was later developed [22]. A last variation used the dynamic conditional correlation model [23] that was developed in [9].

Long Term Generation Algorithm

This chapter develops the Long Term Generation (LTG) algorithm to generate a long-term sequence of wind power outcomes conditioned on sequential short-term forecasts. The benefit of the LTG algorithm over the second approach described in the literature review is that historical wind forecasts can be used, thereby avoiding the difficulty of generating a multivariate time series of forecasts. The benefit over the third approach is that long-term sequences are generated, as opposed to short-term sequences, so that a power system's long-term performance can be

estimated.

The LTG algorithm is a generalization of the marginal-then-joint algorithm. Wind scenarios are generated over a long-term time horizon from time step T_1 to T_2 using historical wind power outcome and forecast data that occurred between time steps $T_1' \leq T_1$ to $T_2' \geq T_2$. The two steps of the marginal-then-joint approach can be accomplished with any of the methods mentioned above, although in this Chapter kernel density estimation is used to create the marginal distributions at each time step $t = T_1, ..., T_2$ and the Gaussian copula approach creates the joint distribution. Finally, an important characteristic of the LTG algorithm is that it can handle cases where some of the historical forecast and outcome data are missing, which is not uncommon in wind data sets.

Joint Distribution Comparison Test

The application of evaluating a power system's performance also motivates the Joint Distribution Comparison (JDC) test developed in this Chapter. It is important to ensure that the wind scenarios are generated from the probability distribution that is conditioned on all of the sequential short-term wind forecasts because the performance of a power system depends on the sequence of both forecasts and outcomes. The developed JDC test is designed to test the hypothesis that the wind scenarios are drawn from the same distribution conditioned on the historical wind forecasts as the the historical wind outcome. This test is different from existing scenario evaluation methods, including the Minimum Spanning Tree rank histogram [36, 30] and Brier Scores [12, 30], which only compare the synthetic wind scenarios to the historical wind without considering the forecast information.

This Chapter is organized as follows. Section 3.4 provides a description of the

notation used throughout the Chapter. The new Long Term Generation (LTG) algorithm is presented in Section 3.5, along with a description of seven variations of the LTG algorithm that are tested. The new Joint Distribution Comparison (JDC) test is developed in Section 3.6. Section 3.7 first describes the method, which we call the Naive Concatenation Method (NCM), of using existing short-term wind generation algorithms to 'piece-together' short-term wind scenarios into a long-term scenario that can be used for simulation. The remainder of Section 3.7 presents computational results and shows a comparison of LTG to NCM. Conclusions are summarized at the end.

3.4 Notation

It is assumed that time is discrete. At time step t the wind state consists of both the wind power at the previous time step as well as the wind power forecast at the next Δ time steps. In this work it is assumed that the wind power forecast i hours in advance is a point forecast. However, everything presented in this Chapter can be generalized to where the forecast is probabilistic, such as a set of quantiles. The wind state can then be represented as a $(\Delta + 1)$-dimensional vector $(w^t, f_1^t, ..., f_\Delta^t)$, where w^t is the historical wind power that occurred at time step t, and $f_i^t, 1 \leq i \leq \Delta$ is the historical wind forecast generated at time t that predicts the wind power at time step $t + i$. For convenience, all of the wind states are normalized so that each wind state $w^t, f_i^t \in [0, 1]$ for all t and each $i = 1, ..., \Delta$. This normalization can be accomplished by dividing the historical wind power values by the total wind power capacity. The wind state time series can be studied as a sample from the stochastic process $\{W^t, F_i^t\}_{t\in\mathbb{N},1\leq i\leq\Delta}$ that has probability measure P. For notation the capitalized W^t and F_i^t are random variables and

the historical data samples are denoted with the lower case w^t or f_i^t, respectively. Finally, let $F^t = (F_1^t, ..., F_\Delta^t)$ be the forecast random vector at time t, and let $f^t = (f_1^t, ..., f_\Delta^t)$ be the historical values.

3.5 Long Term Generation Method

The Long Term Generation method generates long-term synthetic wind scenarios over time horizon T_1 to T_2 conditioned on sequential short-term forecasts. Suppose the historical data begins at time step $T_1' \leq T_1$ and ends at time step $T_2' \geq T_2$. If $P(W^{T_1}, ..., W^{T_2} | F^{T_1'}, ..., F^{T_2'})$ is the conditional distribution of wind outcomes $W^{T_1}, ..., W^{T_2}$ given the sequential wind forecasts $F^{T_1'}, ..., F^{T_2'}$, then the objective of the LTG algorithm is to construct an approximation $\hat{P}$ that has the same conditional distribution, meaning

$$\hat{P}(W^{T_1}, ..., W^{T_2} | F^{T_1'}, ..., F^{T_2'}) \stackrel{d}{=} P(W^{T_1}, ..., W^{T_2} | F^{T_1'}, ..., F^{T_2'}). \tag{3.1}$$

Given $\hat{P}$ and the historical wind forecast samples $f^{T_1'}, ..., f^{T_2'}$, synthetic wind scenarios can be drawn from $\hat{P}(W^{T_1}, ..., W^{T_2} | f^{T_1}, ..., f^{T_2})$ to provide additional wind data sequences on which power systems can be simulated.

Table 3.1: Seven variations of Long-Term Generation method.

Name	Predictor, $X^{t,0}$ (Step (1a))	Window, N (Step (1b))	Covariance, ϵ (Step (2))
X1	(S_1^{t-1})	$2 \cdot 24 \cdot 30$	4.5
X1N2	(S_1^{t-1})	$4 \cdot 24 \cdot 30$	4.5
X4	(S_2^{t-2})	$2 \cdot 24 \cdot 30$	9
X1X4	(S_1^{t-1}, S_4^{t-4})	$2 \cdot 24 \cdot 30$	7
X12	(S_{12}^{t-12})	$2 \cdot 24 \cdot 30$	12
X24	(S_{24}^{t-24})	$2 \cdot 24 \cdot 30$	14
CYCLE	(S_1^{t-1}),(S_2^{t-2}), (S_3^{t-3}),(S_4^{t-4})	$2 \cdot 24 \cdot 30$	7

The Long Term Generation algorithm is presented in Algorithm 1, and it follows the marginal-then-joint distribution approach. In Step 1, for each time step t predictor variables $X^{t,0}$ are defined in order to approximate the marginal distributions $P(W^t|F^{T_1'},...,F^{T_2'})$ by $\hat{P}(W^t|X^{t,0})$. In Step 2, the joint distribution $P(W^{T_1},...,W^{T_2}|F^{T_1'},...,F^{T_2'})$ is approximated by $\hat{P}(W^{T_1},...,W^{T_2}|X^{T_1,0},...,X^{T_2,0})$. Finally, in Step 3 samples are drawn from the approximated joint distribution. The seven LTG variations tested in Section 3.7 are summarized in Table 3.1.

Algorithm 1: Long-Term Generation (LTG)

INPUT: Select time horizon $T_1 < T_2$ over which to generate scenarios, and gather historical data $w^t, f_i^t, i = 1,...,\Delta$ for time steps $t = T_1',...,T_2'$, with $T_1' \leq T_1$ and $T_2' \geq T_2$.

STEP 1: For each time step $t = T_1,...,T_2$ define the predictor variables $X^{t,u}$ as a function

$$X^{t,u} = h^t(...,F^{t+u-1},F^{t+u},F^{t+u+1},...) \tag{3.2}$$

of the forecast random variables. The t superscript denotes that the function h^t can be a function of the time step, and u is an integer specifying an offset; the purpose for this notation is explained more clearly in Steps (1a) and (1b). The historical sample of the random variable $X^{t,u}$ will be denoted with the lowercase $x^{t,u} = h^t(...,f^{t+u-1},f^{t+u},f^{t+u+1},...)$.

STEP 1a: For each time step $t = T_1,...,T_2$ define the predictor variables as shown in Equation (3.2) by creating a function h^t of the wind forecasts.

Six different predictor variables are tested. In LTG variations X1 and X1N2 the single predictor variable is $X^{t,u} = F_1^{t+u-1}$. Thus, for each $t = T_1, ..., T_2$ the distribution $\hat{P}(W^t|F_1^{t-1})$ will be estimated, which is the distribution of wind power at time t conditioned on the previous time step's one time step ahead point forecast. Variations X4, X12, and X24 are analogous, except they use the 4, 12 and 24 time step ahead forecast. In X1X4 the two predictor variables are $X^{t,u} = (F_1^{t+u-1}, F_4^{t+u-4})$, i.e. the one and four time step ahead forecasts. Finally, in LTG variation CYCLE the predictor variables depend on the time step. At time step t the single predictor variable is the $(mod(t, 4) + 1)$ time step ahead forecast. This variation is tested so as to provide a comparison to the Naive Concatenation Methods developed in Section (3.7.1).

STEP 1b: For each $t = T_1, ..., T_2$ gather historical samples drawn from $P(W^t|X^{t,0})$.

If no assumptions are made about P then the single historical data sample drawn from $P(W^t|X^{t,0})$ is $(x^{t,0}, w^t)$, which is insufficient for constructing the estimated distribution $\hat{P}(W^t|X^{t,0})$. So, it will be assumed that P is slowly time varying, meaning $P(W^{t+u}|X^{t,u})$ has the same distribution as $P(W^t|X^{t,0})$ when $|u|$ is small. Therefore, by selecting some even integer $N > 0$, the data samples used to estimate $P(W^t|X^{t,0})$ are

$$D(t) = \{(x^{t,u}, w^{t+u})\}_{-N/2 \leq u \leq N/2}. \tag{3.3}$$

As described in Section (3.7.2) the wind data has hourly time steps, so the $N = 4{\cdot}24{\cdot}30$ samples in variation X1N2 contains approximately four months' worth of data, while the $N = 2 \cdot 24 \cdot 30$ samples in all other variations is approximately two months' worth.

STEP 1c: For each $t = T_1, ..., T_2$ use the training data $D(t)$ generated in Step (1b) to estimate the conditional distribution $P(W^t|X^{t,0})$.

All LTG variations use kernel density estimation (KDE) with a beta kernel [14] to estimate the marginal distributions, although any of the methods described in the Introduction could be used. The implemented KDE algorithm makes two modifications to the algorithm described in [6, 7]. To fit a KDE model to a data set the KDE parameters, called kernel scalings, are optimized using a cross validation objective function. This is a computationally expensive objective function that scales as N^2. The first difference is that the Global Metric Stochastic Radial Basis Function (GMSRBF) algorithm, which is designed for computationally expensive functions, was applied to optimize the scalings [29]. GMSRBF found the optimal scalings faster than local optimization methods.

The second modification assumes the optimal kernel scalings are slowly time varying, which allows for two computational savings techniques. First, it is assumed the scalings are constant over a period of one week, so that the optimization problem only needs to be solved once per week as opposed to every time step. Second, it is assumed that the optimal scalings at the next week are in the neighborhood of the scalings from the previous week, thereby making the domain of the optimization problem smaller. Specifically, the search space for each kernel scaling is between 10^{-8} and 10^2, which is optimized in log-space. The neighborhood of a scaling $10^{-8} \leq x \leq 10^2$ is defined as $[10^{log_{10}(x)-1}, 10^{log_{10}(x)+1}]$, which reduces the search space for a single scaling by 80%.

STEP 2: Construct the estimated joint distribution
$\hat{P}(W^{T_1}, W^{T_1+1}, ..., W^{T_2}|X^{T_1,0}, X^{T_1+1,0}, ..., X^{T_2,0})$.

All variations in Table 3.1 use the Gaussian copula method with an exponential covariance function, as described in [26]. After transforming each of the marginal distributions into a standard normal distribution, the joint distribution is uniquely defined by the covariance matrix Σ, given by

$$\Sigma_{i,j} = exp(-|i-j|/\epsilon). \tag{3.4}$$

Section (3.7.3) describes how the parameter ϵ is selected.

STEP 3: Draw long term scenarios.

Calculate the historical predictor variables $x^{t,0} = h^t(..., f^{t-1}, f^t, f^{t+1}, ...)$ for $t = T_1, ..., T_2$. Generate scenarios over time horizon T_1 to T_2 by randomly drawing samples from the joint distribution $\hat{P}(W^{T_1}, W^{T_1+1}, ..., W^{T_2} | x^{T_1,0}, x^{T_1+1,0}, ..., x^{T_2,0})$. The method of drawing scenarios using the Gaussian copula method is described in, e.g., [28].

3.5.1 Handling Missing Historical Data

There is often missing historical data over the time horizon T_1' to T_2'. If there is missing data when constructing the data sets $\mathcal{D}(t)$ in Step (1b) then these missing points can be ignored, and in Step (1c) there will just be fewer data points on which the marginal distribution is estimated.

However, in Step (3) there will be instances where forecast data is missing that is required to calculate the historical predictor variables $x^{t,0} =$

$h^t(..., f^{t-1}, f^t, f^{t+1}, ...)$. If $x^{t,0}$ cannot be calculated then the marginal distribution $\hat{P}(W^t|x^{t,0})$ cannot be estimated, and then the joint distribution $\hat{P}(W^{T_1}, ..., W^{T_2}|x^{T_1,0}, ..., x^{T_2,0})$ cannot be estimated. In these cases a new predictor variable function $\hat{h}^t$ must be selected such that even with the missing data the historical predictor value $\hat{x}^{t,0} = \hat{h}^t(..., f^{t-1}, f^t, f^{t+1}, ...)$ exists. The predictor function $\hat{h}^t$ will then replace h^t in the Step 1 of the LTG method, and the algorithm can proceed as usual.

A general method, which will be called 'backshifting', can be used that selects a new predictor function $\hat{h}^t$ based on the desired function h^t such that $\hat{x}^{t,0}$ exists when, due to missing historical data, $x^{t,0}$ does not. For a positive integer B, the main idea of backshifting is to shift the predictor variables by using the forecasts generated B time steps earlier and a forecast horizon B more time steps in advance. More technically, for positive integer B define

$$F^t(B) = (F^t_{B+1}, F^t_{B+2}, ..., F^t_{\Delta}) \tag{3.5}$$

to be the random vector of forecasts generated at time t that only includes the forecasts that are more than B time steps ahead. Then the backshifting method is to select the smallest B such that by defining

$$\hat{h}^t(..., F^{t-1}, F^t, F^{t+1}, ...) = h^t(..., F^{t-1-B}(B), F^{t-B}(B), F^{t+1-B}(B), ...) \tag{3.6}$$

the predictor value $\hat{x}^{t,0} = \hat{h}^t(..., f^{t-1}, f^t, f^{t+1}, ...)$ exists.

For example, suppose for each t that $h^t(..., F^{t-1}, F^t, F^{t+1}, ...) = F^{t-1}_1$, so that the predictor variable is the previous time step's one time step ahead forecast. Also, suppose the historical forecast f^t exists at time step $t = T$ but it is missing at time step $t = T + 1$. Then at time step $T + 2$ the predictor value would be

f_1^{T+1}, but this historical data point is missing. Using the backshifting method with $B = 1$ the new predictor variable is instead

$$\hat{h}^{T+2}(..., F^{t-1}, F^t, F^{t+1}, ...) = h^{T+2}(..., F^T(1), F^{T+1}(1), F^{T+2}(1), ...) = F_2^T, \quad (3.7)$$

which is the two time step ahead forecast generated two time steps ago, and so the historical predictor variable f_2^T exists.

3.5.2 Existing Evaluation Methods

Two common methods for evaluating synthetic wind scenarios are the Minimum Spanning Tree (MST) rank histogram [36, 30] and Brier Scores [12, 30]. Both of these methods compare the historical wind outcomes w^t to the synthetic wind scenarios $\omega^t(j)$, and they do not consider the statistical behavior of the scenarios when conditioned on wind forecasts.

The MST rank histogram test measures scenario calibration, which is the statistical similarity between the historical outcomes and the scenarios' distribution [16]. The MST rank histogram test has previously only been applied to short-term scenarios. Suppose the synthetic scenarios are of length H time steps. Let the H-dimensional historical wind outcome be w, and let the N_S scenarios, which are also H-dimensional, be $\omega(j), j = 1, ..., N_S$. Let L_0 be the minimum spanning tree length of the N_S wind scenario data points, and let $L_i, i = 1, ..., N_S$ be the minimum spanning tree length of the data points $\{w, \omega(j) | j = 1, ..., N_S, j \neq i\}$. Finally, order the $L_i, i = 0, ..., N_S$ from smallest to largest and calculate the rank of L_0, which is between 1 and $N_S + 1$. If the wind scenarios are drawn from the same distribution as the historical wind then each rank is equally likely. If scenarios are generated over multiple time intervals, each over H time steps, then the

histogram of ranks should be uniform. Many high ranks indicates the scenarios are over dispersed, and many low ranks means the scenarios are either under dispersed or biased.

The MST rank histogram test is slightly modified to be applied to long-term scenarios. The long-term scenarios and corresponding historical wind are first partitioned into intervals of H time steps. The MST test can then be applied as usual.

Brier scores are a verification method that measure the similarity between scenarios and the historical outcome based on a set of predefined events. The Brier score is the mean over all time steps of the absolute difference between an indicator (either 0 or 1) that the historical event occurred and the probability the event occurred in the scenarios. Brier scores are bounded between 0 and 1, with 0 indicating the scenarios are statistically similar. The same two events are used that are defined in the reference, which are a ramp-up event, defined by whether the wind increased more than a threshold ξ in the previous κ time steps, and an analogous ramp-down event.

3.6 Joint Distribution Comparison

The Joint Distribution Comparison (JDC) test is developed to assess the statistical similarity between the conditional distribution

$$P(W^{T_1}, ..., W^{T_2} | F^{T_1'}, ..., F^{T_2'}) \tag{3.8}$$

and the approximated conditional distribution

$$\hat{P}(W^{T_1}, ..., W^{T_2} | F^{T_1'}, ..., F^{T_2'}). \tag{3.9}$$

The main idea is to compare the joint distribution of pairs of a forecast and an outcome random variable. That is, for integer L and $\delta \in \{1, ..., \Delta\}$ the Joint Distribution Comparison tests the null hypothesis

$$\begin{aligned} \mathcal{H}_0(L, \delta) : \text{The joint distribution } P(F_\delta^{t-L}, W^t) \text{ is the} \\ \text{same as } \hat{P}(F_\delta^{t-L}, W^t) \text{ for all } t = T_1, ..., T_2. \end{aligned} \tag{3.10}$$

Notice that the null hypothesis is a statement over all time steps $t = T_1, ..., T_2$.

For a given L and δ the JDC test uses the two-sample test in [17] to test Hypothesis (3.10) to determine whether the historical data set $\{(f_\delta^{t-L}, w^t)\}_{t=T_1}^{T_2}$ and the synthetic data set $\{(f_\delta^{t-L}, \omega^t(j)) : j = 1, ..., N_S\}_{t=T_1}^{T_2}$ could have been drawn from the same distribution. However, the two-sample test assumes that 1) for each data set all the samples were drawn from a single distribution, and 2) the samples in each data set are all independent.

To satisfy assumption (1) it is assumed that both P and $\hat{P}$ are slowly time varying, meaning that for small $|u|$ the distributions $P(F_\delta^{t-L+u}, W^{t+u})$ are the same as $P(F_\delta^{t-L}, W^t)$, and similarly for $\hat{P}$. So, define partition time steps T_i^P such that $T_1 = T_1^P < T_2^P < \cdots < T_{N_P+1}^P = T_2 + 1$, and define the historical partitioned data

$$\mathcal{D}_i = \{(f_\delta^{t-L}, w^t) : T_i^P \leq t < T_{i+1}^P\} \tag{3.11}$$

and the synthetic partitioned data for each synthetic scenario $j = 1, ..., N_S$,

$$\hat{\mathcal{D}}_{i,j} = \{(f_\delta^{t-L}, \omega^t(j)) T_i^P \leq t < T_{i+1}^P\} \tag{3.12}$$

for each $i = 1, ..., N_P$. For each of these datasets it can then be assumed that the data were all drawn from the same distribution.

The second assumption, that the data samples are independent, is likely not true because wind is strongly correlated. To satisfy this assumption the data sets in Equations (3.11) and (3.12) can be randomly partitioned into K sets. For some sufficiently large K it can be assumed that the samples in each of these twice-partitioned sets are independent.

Altogether, the JDC test is in Algorithm 2. In the results shown in Section 3.7 the partitions are selected to contain three consecutive months of data, $N_S = 100$, and $K = 50$.

Algorithm 2: Joint Distribution Comparison (JDC)

---

1. Let N_S be the number of synthetic wind scenarios. Select a positive integer K that defines the number of partitions.
2. Define the joint random variable F_δ^{t-L} by selecting the lead time L and a forecast horizon δ. The null hypothesis being tested is stated in Hypothesis (3.10).
3. Partition the time horizon T_1 to T_2 into N_P intervals by defining $T_1 = T_1^P < T_2^P < \cdots < T_{N_P+1}^P = T_2 + 1$. It is assumed that for each $i = 1, ..., N_P$ the joint distributions $P(F_\delta^{t-L}, W^t)$ are equal for $T_i^P \leq t < T_{i+1}^P$. Similarly it is assumed the approximated joints distributions $\hat{P}(F_\delta^{t-L}, W^t)$ are equal for $T_i^P \leq t < T_{i+1}^P$.
4. For each $i = 1, ..., N_P$ gather historical and synthetic data sets defined in Equations (3.11) and (3.12).

5. For each time horizon $i = 1, ..., N_P$ and each scenario $j = 1, ..., N_S$ perform the following process:

   (a) Randomly partition the historical dataset $\mathcal{D}_i$ into K equal sized sets $\mathcal{D}_{i,k}, k = 1, ..., K$.

   (b) Randomly partition the synthetic dataset $\hat{\mathcal{D}}_{i,j}$ into K equal sized sets $\hat{\mathcal{D}}_{i,j,k}, k = 1, ..., K$.

   (c) For each partition $k = 1, ..., K$ use the nearest neighbor two-sample test [17] to calculate the p-value that datasets $\mathcal{D}_{i,k}$ and $\hat{\mathcal{D}}_{i,j,k}$ were drawn from the same distribution. Set this p-value to $p_{i,j,k}$.

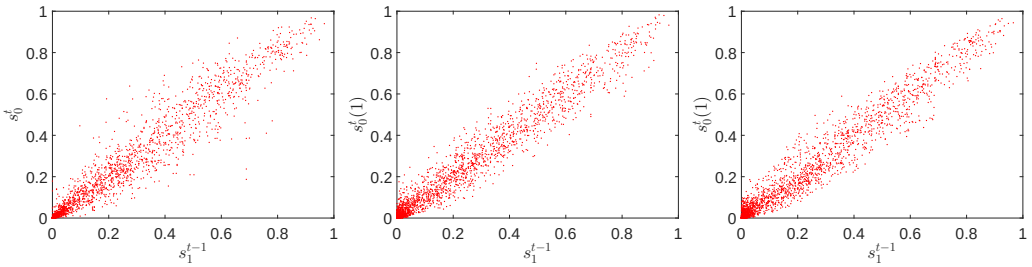
6. Combine all the p-values together using Fisher's method by calculating the statistic

$$\chi^2 = -2 \sum_{i=1}^{N_P} \sum_{j=1}^{N_S} \sum_{k=1}^{K} log(p_{i,j,k}). \tag{3.13}$$

   This statistic has a χ^2 distribution with $2N_P N_S K$ degrees of freedom [15]. The null hypothesis in Equation (3.10) can be tested by comparing the statistic with the critical 95% value.

3.6.1 Application of JDC on Simulated Data

The ability of our test to differentiate samples drawn from different distributions is demonstrated on a set of synthetic data. First, a distribution P'_ϵ is created from which synthetic 'historical' data can be drawn. Then, P'_ϵ is given bias $\delta\mu$ and variance scaling δv to create a second perturbed distribution $\hat{P}'_{\hat{\epsilon},\delta\mu,\delta v}$ from which



(a) Historical joint distribution data (b) Simulated distribution $P'_{\epsilon=0.1}$ (c) Simulated distribution $P'_{\epsilon=10}$

Figure 3.1: Qualitatively showing the simulated distributions (b) and (c) are similar to the historical BPA data set (a).

the scenarios are drawn. The JDC test is then applied to these synthetic scenarios to determine if P'_ϵ and $P'_{\hat{\epsilon},\delta\mu,\delta v}$ are different, in order to illustrate the effectiveness of the JDC test.

In order to make this demonstration of JDC more realistic, the distribution P'_ϵ is based on the data set from Bonneville Power Administration described in Section 3.7.2. The time horizon in this test is from T_1 on August 1, 2013 to T_2 on October 31, 2013, and the data consists of hourly wind forecasts and wind observations.

Creating Distribution P'_ϵ

The distribution P'_ϵ is constructed using the marginal-then-joint approach. The model for P'_ϵ is chosen so as to be simple and yet flexible enough to have similar characteristics as the BPA data set. At each time step t the marginal distribution $P'_\epsilon(W^t|f_1^{t-1})$ is defined to be a beta distribution

$$P'_\epsilon(W^t = r|f_1^{t-1}) = Beta(r; \alpha(f_1^{t-1}), \beta(f_1^{t-1})), \tag{3.14}$$

where $Beta(r;\alpha,\beta)$ is the probability density of a beta distribution with parameters α and β at $r \in [0,1]$. The parameters $\alpha(f_1^{t-1})$ and $\beta(f_1^{t-1})$, which are functions of

the BPA historical data point f_1^{t-1}, are chosen so that, with parameters $a \in [-1, 1]$ and $v \in (0, 1]$, the mean of the of the beta distribution is

$$\mu(f_1^{t-1}) = af_1^{t-1} + \frac{1-a}{2} \tag{3.15}$$

and the variance is

$$\sigma^2(f^{t-1}) = v(\mu(f_1^{t-1}) - \mu(f_1^{t-1})^2). \tag{3.16}$$

The mean of the beta distribution is a linear function of the forecast such that $\mu(0.5) = 0.5$ for every a. The variance of the beta distribution is the function in Equation (3.16) because the maximum variance of a beta distribution with mean μ is $\mu - \mu^2$.

The mean and variance of a beta distribution with parameters $\alpha(f_1^{t-1})$ and $\beta(f_1^{t-1})$ are

$$\mu = \frac{\alpha(f_1^{t-1})}{\alpha(f_1^{t-1}) + \beta(f_1^{t-1})} \tag{3.17}$$

and

$$\sigma^2 = \frac{\alpha(f_1^{t-1})\beta(f_1^{t-1})}{(\alpha(f_1^{t-1}) + \beta(f_1^{t-1}))^2(\alpha(f_1^{t-1}) + \beta(f_1^{t-1}) - 1)}. \tag{3.18}$$

Inverting Equations (3.16), (3.17), and (3.18) shows that the unique values for parameters $\alpha(f_1^{t-1})$ and $\beta(f_1^{t-1})$ are

$$\alpha(f_1^{t-1}) = \mu(f_1^{t-1})\Big(\frac{1}{v} - 1\Big) \tag{3.19}$$

and

$$\beta(f_1^{t-1}) = \alpha(f_1^{t-1})\Big(\frac{1}{\mu(f_1^{t-1})} - 1\Big). \tag{3.20}$$

Altogether, Equations (3.14), (3.19), and (3.20) define the marginal distribution $P'_\epsilon(W^t|f_1^{t-1})$ with parameters a in Equation (3.15) and v in Equation (3.20).

The joint distribution is created using the Gaussian copula approach with an exponential covariance matrix, where the covariance between marginal distributions at time steps t and u is $exp(-|t-u|/\epsilon)$. The two values of the covariance parameter ϵ that are chosen are $\epsilon = 10^{-1}$ and $\epsilon = 10$; in the first case the marginal distributions are effectively mutually independent, and in the second they are highly correlated.

The values for the two parameters a and v in Equations (3.15) and (3.16) are determined using Maximum Likelihood Estimation with joint distribution parameter set to $\epsilon = 0$. The MLE is performed over the described historical BPA data, and the resulting values are $a = 0.98$ and $v = 0.025$. Figure 3.1 demonstrates that at least qualitatively the developed simulated model (middle and right) produces a distribution similar to the historical dataset (left). The covariance parameter is set to 0.1 and 10 in the middle and right subfigures respectively.

Creating biased and scaled distributions $\hat{P}'_{\epsilon,\delta\mu,\delta v}$

The perturbed distributions $\hat{P}'_{\hat{\epsilon},\delta\mu,\delta v}$ are created using the bias $\delta\mu$ and variance scaling δv. The distribution of W^t is

$$P'_{\hat{\epsilon},\delta\mu,\delta v}(W^t = r|f_1^{t-1}) = Beta(r; \alpha'(f_1^{t-1}, \delta\mu, \delta v), \beta(f_1^{t-1}, \delta\mu, \delta v)), \tag{3.21}$$

where

$$\alpha'(f_1^{t-1}, \delta\mu, \delta v) = (\mu(f_1^{t-1}) + \delta\mu)\Big(\frac{1}{v\delta v} - 1\Big) \tag{3.22}$$

and

$$\beta'(f_1^{t-1}, \delta\mu, \delta v) = \alpha'(f_1^{t-1}, \delta\mu, \delta v)\Big(\frac{1}{\mu(f_1^{t-1}) + \delta\mu} - 1\Big). \tag{3.23}$$

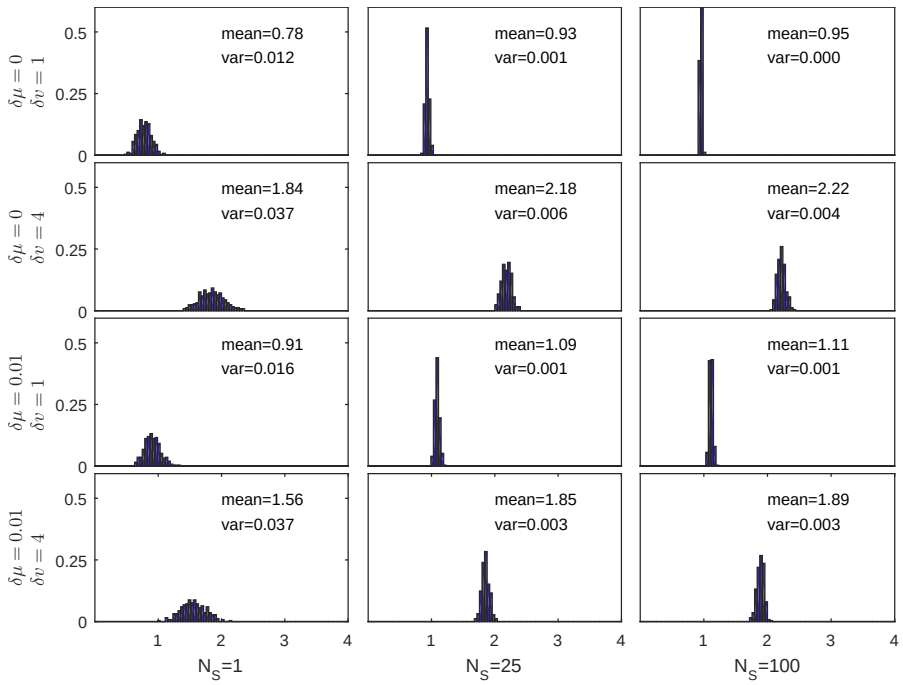
Notice that δv is a multiplicative factor and, since $v \in (0, 1]$ and it is required that $v\delta v \in (0, 1]$, this means that $\delta v \in (0, 1/v]$. Also, because the mean of a beta distribution must be between 0 and 1, the bias $\delta\mu$ is constrained by $\delta\mu \in [(a-1)/2, (1-a)/2]$. An unscaled and unbiased perturbed distribution is obtained with $\delta v = 1$ and $\delta\mu = 0$.

Statistical Test Analysis

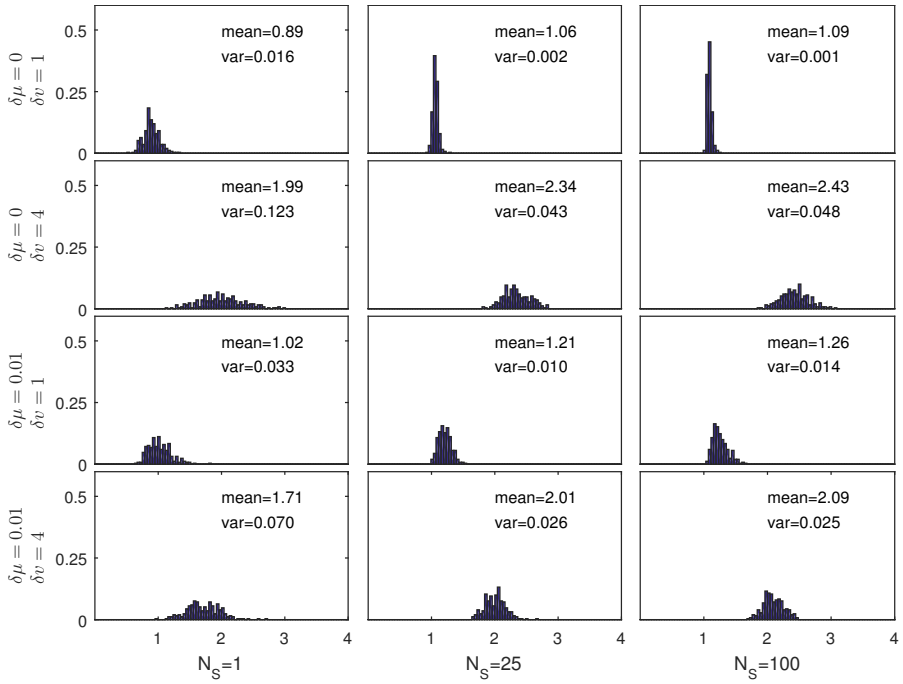
The first two tests explore the impact of the two parameters N_S and K used in steps 1 and 5 of the JDC test, with $N_P = 1$ held fixed. The histograms in Figures (3.2) and (3.3) are created using the same procedure: First, P'_ϵ is selected, with $\epsilon = \hat{\epsilon} = 0.1$ in subfigures (a) and $\epsilon = \hat{\epsilon} = 10$ in (b). Second, $\hat{P}'_{\hat{\epsilon},\delta\mu,\delta v}$ is selected, where $\delta\mu$ and δv are specified to the left of each row of histograms in the Figure. Third, a single historical sample $w^{T_1}, ..., w^{T_2}$ is drawn from P'_ϵ and N_S scenarios $\omega^{T_1}(j), ..., \omega^{T_2}(j), j = 1, ..., N_S$ are drawn from $\hat{P}'_{\hat{\epsilon},\delta\mu,\delta v}$. In Figure (3.2) $K = 50$ is held fixed and N_S is specified at the bottom of each column of histograms, and in Figure (3.3) $N_S = 100$ is fixed and K is specified at the bottom. And fourth, the JDC test in Algorithm 2 is applied to these datasets $\mathcal{D}_1 = \{(f_1^{t-1}, w^t) : T_1 \leq t \leq T_2\}$ and $\mathcal{D}_{1,j} = \{(f_1^{t-1}, \omega^t(j)) : T_1 \leq t \leq T_2\}$ to obtain the chi-squared test statistic χ^2 in Equation (3.13). The chi-squared statistic has $2KN_S$ degrees of freedom, so to provide consistent comparisons when N_S and K are varied the chi-squared values are all normalized by the 95% critical value $\chi^2_{95,2KN_S}$. All histograms therefore plot the critical value ratio

$$\chi^2_{ratio} = \frac{\chi^2}{\chi^2_{95,2KN_S}}. \tag{3.24}$$

If $\chi^2_{ratio} \leq 1$ then the hypothesis that the distributions are the same is not rejected at the 95% confidence level. If $\chi^2_{ratio} > 1$ then the hypothesis is rejected. This



(a) covariance parameters $\epsilon = \hat{\epsilon} = 0.1$



(b) covariance parameters $\epsilon = \hat{\epsilon} = 10$

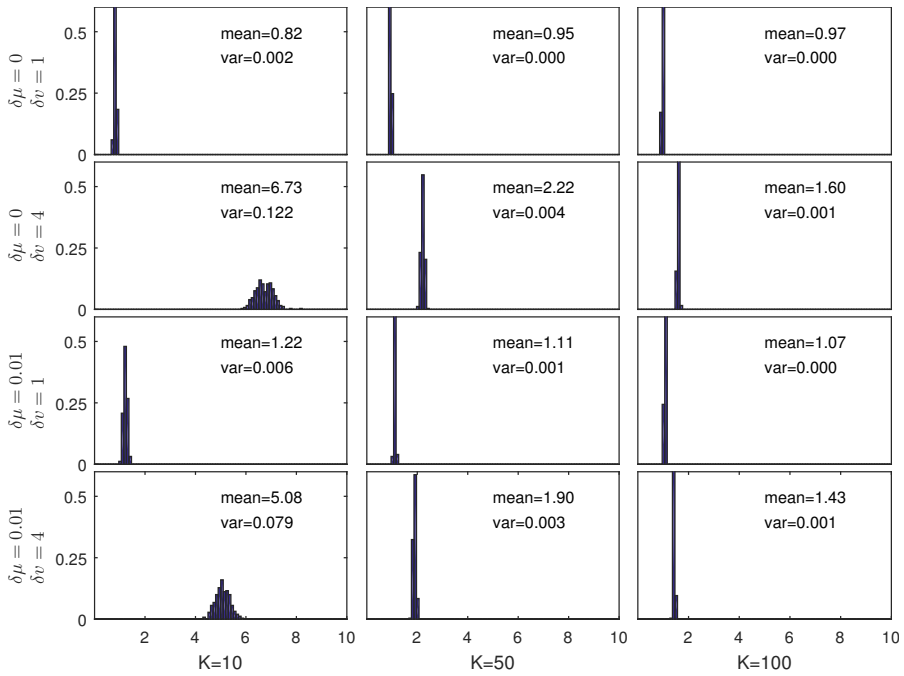
Figure 3.2: Examining the impact of increasing the number of scenarios N_S on JDC, with $K = 50$. The x-axis is χ^2_{ratio} in Equation (3.24), and the y-axis is the histogram relative frequency. In each row of three plots the perturbed distribution has a different bias and scaling, shown on left.

process is repeated 250 times, so each histogram consists of 250 data points of χ^2_{ratio}.

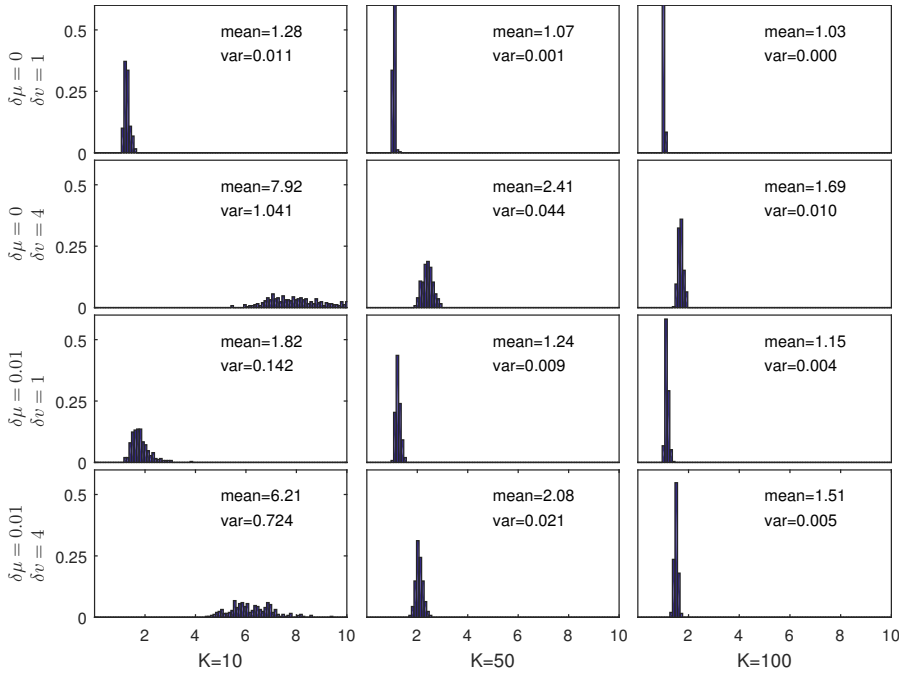
Figure 3.2 shows that as N_S increases from 1 (first column of histograms in both of the two subfigures) to 25 (second column) to 100 (third column) the variance of the histogram almost always decreases. However, the variances are much smaller in subfigure (a), where the marginal distributions of P'_ϵ and $\hat{P}'_{\hat{\epsilon},\delta\mu,\delta v}$ are nearly independent because the covariance parameter is small, than in subfigure (b). This indicates that the reliability of the JDC test depends on the whether the samples in $\mathcal{D}_i$ and $\mathcal{D}_{i,j}$ are independent.

An important observation is that the JDC test can differentiate when samples are drawn from different distributions. Comparing the four histograms in any column shows that the mean χ^2_{ratio} obtained when $\delta\mu = 0$ and $\delta v = 1$ (i.e. the perturbed distribution is not biased and not scaled) is always less than the mean χ^2_{ratio} obtained when the perturbed distribution is biased or scaled. This is the exact behavior that is desired because it shows that the JDC test can correctly identify that the non-biased and non-scaled perturbed distribution is more similar to the distribution than a biased and/or scaled perturbed distribution. The benefit of increasing N_S is that the variance decreases, which helps the JDC test more easily determine if a perturbed distribution is biased and/or scaled. When $\epsilon = \hat{\epsilon} = 0.1$ the mean of the histograms when the perturbed distribution is unbiased and unscaled is less than 1, as would be expected. However, when $\epsilon = \hat{\epsilon} = 10$ the mean is slightly larger than 1, which again indicates that the test depends on the independence of the data.

Figure 3.3 shows the results of an analogous test, except that K is varied from $K = 10, 50$, and 100, and $N_S = 100$ is held fixed. Again, a total of 250 trials are



(a) covariance parameters $\epsilon = \hat{\epsilon} = 0.1$



(b) covariance parameters $\epsilon = \hat{\epsilon} = 10$

Figure 3.3: Examining the impact of increasing the number of scenarios K on JDC, with $N_S = 100$. The x-axis is χ^2_{ratio} in Equation (3.24), and the y-axis is the histogram relative frequency. In each row of three plots the perturbed distribution has a different bias and scaling, shown on left.

performed to obtain each histogram. As K becomes larger the mean χ^2_{ratio} value becomes closer to 1, which is likely because when K is small the datasets $\mathcal{D}_{i,j,k}$ and $\hat{\mathcal{D}}_{i,j,k}$ in Equations (3.11) and (3.12) have more data points, and so for small K the JDC test can more definitively reject or fail to reject the null hypothesis as compared to when K is large.

K can be chosen by analyzing the impact on the ability of the JDC test to determine if samples are drawn from different distributions. Specifically, if $\hat{P}'_{\epsilon,\delta\mu,\delta v}$ is a perturbed distribution that is biased and/or scaled and $\hat{P}'_{\epsilon,0,1}$ is the non-biased and non-scaled perturbed distribution, the JDC test should be able to show that that $\hat{P}'_{\epsilon,\delta\mu,\delta v}$ is more different from P'_ϵ than $\hat{P}'_{\epsilon,0,1}$ is from P'_ϵ. By setting $mean_{K,\epsilon,\delta\mu,\delta v}$ and $var_{K,\epsilon,\delta\mu,\delta v}$ to be the mean and variance of the χ^2_{ratio} of the histograms, which are shown next to each histogram in the Figure (3.3), the z-score

$$z_{K,\epsilon,\delta\mu,\delta v} = \frac{mean_{K,\epsilon,\delta\mu,\delta v} - mean_{K,\epsilon,0,1}}{\sqrt{var_{K,\epsilon,\delta\mu,\delta v} + var_{K,\epsilon,0,1}}} \tag{3.25}$$

is a measure of how accurately the JDC test determines whether a biased and/or scaled perturbed distribution is different from the distribution. A larger z-score indicates the JDC test is more accurate.

Table 3.2 shows the z-scores. In almost every case $K = 10$ yields the smallest, or worst, scores. In most cases $K = 50$ yields the largest, or best, scores, and so $K = 50$ is selected. However, the change in z-scores as K is varied is relatively small and appears to have at most a modest impact on the JDC test.

Finally, Figure 3.4 shows the mean JDC value as a function of the bias and scaling of the perturbed distribution. The same procedure described above is repeated for varying biases and scalings, but now with $N_S = 100$ and $K = 50$

Table 3.2: Determining the ability of the JDC test to identify if a perturbed distribution is biased and/or scaled for different values of K. Larger z-scores, defined in Equation (3.25), indicate the JDC test can more reliably distinguish when samples are drawn from different distributions.

	$K = 10$	$K = 50$	$K = 100$
$z_{K,\epsilon=0.1,\delta\mu=0,\delta v=4}$:	16.8098	19.5983	18.8808
$z_{K,\epsilon=0.1,\delta\mu=0.01,\delta v=1}$:	4.5874	5.0535	4.7114
$z_{K,\epsilon=0.1,\delta\mu=0.01,\delta v=4}$:	14.9830	17.4914	16.2254
$z_{K,\epsilon=10,\delta\mu=0,\delta v=4}$:	6.4650	6.2723	6.4114
$z_{K,\epsilon=10,\delta\mu=0.01,\delta v=1}$:	1.3824	1.6200	1.6392
$z_{K,\epsilon=10,\delta\mu=0.01,\delta v=4}$:	5.7484	6.7897	6.5599

held fixed. For each bias and scaling 250 trials are performed and are used to estimate the mean JDC χ^2_{ratio}. Figure 3.4 shows contour plots of this mean ratio. The dot in the center of each plot shows where the bias is $\delta\mu = 0$ and the scaling is $\delta v = 1$. The four contour plots show the results for the different covariance parameter values.

The JDC test can always determine that the unbiased and unscaled perturbed distribution is most similar to P'_ϵ. This is because the mean χ^2_{ratio} value increases as either the bias becomes more non-zero or the scaling becomes farther from 1. Additionally, in the two cases where $\epsilon = 0.1$ the JDC test yields a mean χ^2_{ratio} value less than 1 when comparing the unbiased and unscaled perturbed distribution $\hat{P}'_{\hat{\epsilon},\delta\mu=0,\delta v=1}$ to P'_ϵ, exactly as would be expected. When $\epsilon = 10$ the mean value is slightly larger than 1, which indicates that if the data is highly correlated then the JDC test may have a higher false negative rate than would be expected.

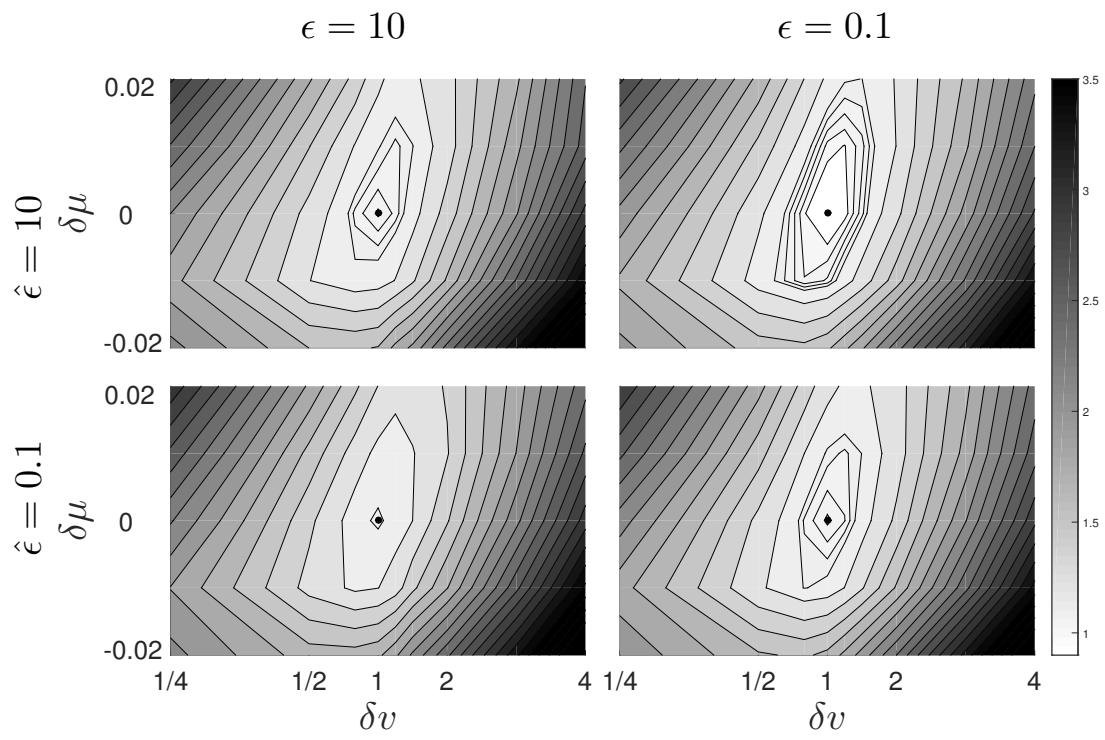


Figure 3.4: Showing that the JDC method can correctly distinguish between similar and dissimilar distributions. The x-axis is the scaling, and the y-axis is the bias. Contours show values of the mean χ^2_{ratio} in Equation (3.24).

3.7 Computational Results

3.7.1 Naive Concatenation Method

As a comparison to LTG, we also develop and test the Naive Concatenation Method (NCM) of generating long-term wind scenarios. The first step of NCM is to select some desired horizon length $\hat{\Delta}$ less than the forecast horizon, i.e. $\hat{\Delta} \leq \Delta$. If the algorithm is at time step t, select the smallest $d \geq \hat{\Delta}$ such that the historical wind forecast at time step $t + d$ exists. Use a short-term generation algorithm, e.g. [28, 22, 23], to generate a wind scenario over time steps $t + 1, ..., t + d$ conditioned on the forecast generated at time t. Next, jump to time step $t + d$ and repeat the

Table 3.3: Three variations of Naive Concatenation Method.

Name	Forecast horizon, $\hat{\Delta}$	Covariance Parameter, ϵ
NCM4	4	14
NCM12	12	13
NCM24	24	13

process. Finally, concatenate these scenarios together to yield a long-term scenario. Notice that consecutive sequences of short-term scenarios are independent.

All NCM variations, listed in Table 3.3, use the marginal-then-joint approach to generate the short-term scenarios of length $\hat{\Delta}$. KDE is used to estimate the marginal distributions $P(W^{t+k}|F_k^t)$ for each $k = 1, ..., d$, and a Gaussian copula with an exponential covariance matrix, with parameter ϵ, creates the joint distribution.

Notice that NCM4, which sets $\hat{\Delta} = 4$, uses the same predictor variables as the LTG variation CYCLE to construct each marginal distribution. The difference is that in NCM4 marginal distributions at time steps i and j are independent whenever $|i - j| > 4$, which is unrealistic, whereas in CYCLE all marginal distributions have a nonzero covariance.

3.7.2 Bonneville Power Administration Dataset

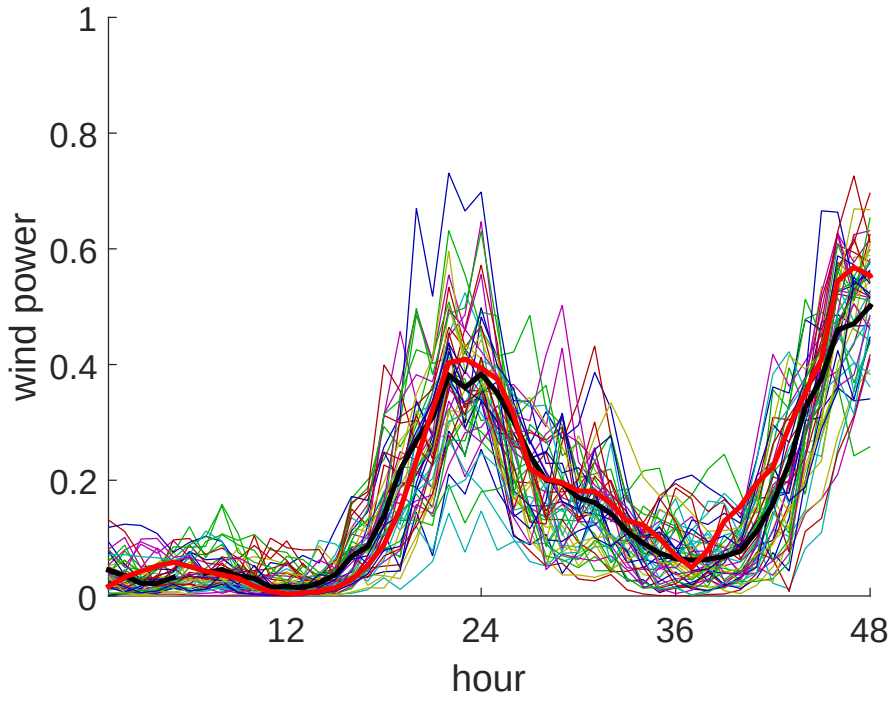
The Bonneville Power Administration provides almost 30% of the electric power consumed in the US Pacific Northwest, and the capacity of the wind power it markets is almost 5000 MW. BPA makes the historical wind forecast and outcome data publicly available [1]. For this study the scenarios were generated with a length of one year, from T_1 = August 1, 2013 to T_2 = August 1, 2014. Time steps are in hours, so the scenarios are $8,760$ time steps long. The historical

data was selected from T_1' = April 3, 2013 to T_2' = December 18, 2014 because the installed wind capacity remained constant over this time period [2]. If wind turbines were added or removed the probability distribution P may have been significantly altered, which could make analysis more difficult. Available forecast information includes Minimum, Average, and Maximum point forecasts, but only the Average forecast is used as the point predictions. There is missing forecast and outcome data, but this is handled as described in Section 3.5.1. Additionally, there are some erroneous repeated forecasts, where $f_i^t = f_i^{t+1}$ for each $i = 1, ..., \Delta$, and these repeated forecasts are removed and treated as missing data. The data were all normalized using the information in [2].

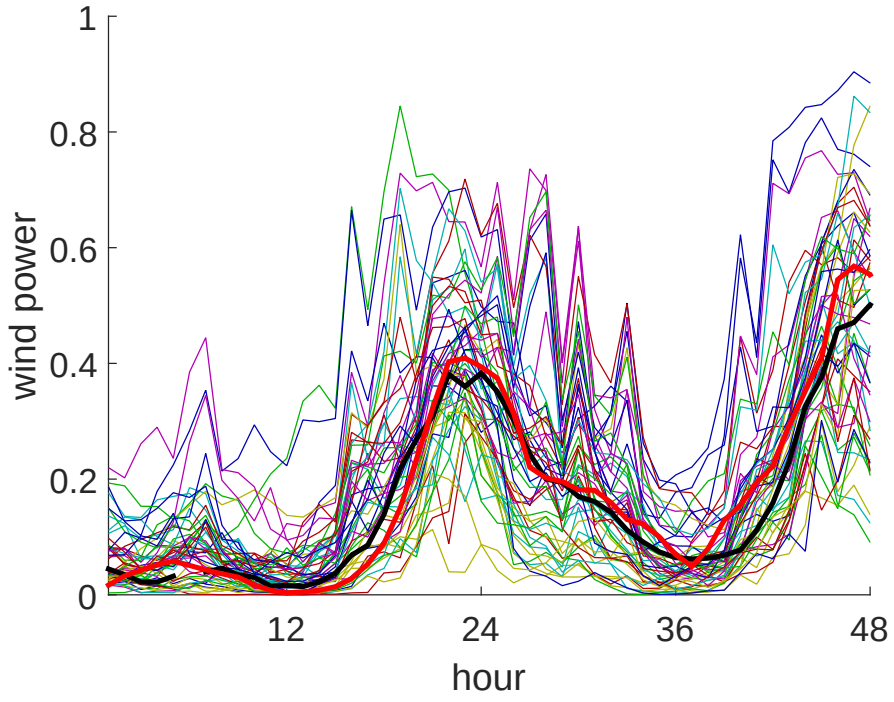
3.7.3 Results

Each of the seven LTG and three NCM variations were run twice, and in each trial 100 synthetic wind scenarios were generated. Four examples of wind scenarios are shown in Figures 3.5 and 3.6 generated by variations X1, X24, NCM4, and NCM24.

The MST rank histogram test is applied by splitting the wind scenarios and historical wind into 365 intervals of $H = 24$ time steps (hours). The MST was performed twice, once on each set of 50 scenarios, and the rank histograms were summed. This yields 51 possible ranks (bins) with an expected value of $365/51 \cdot 2 \approx 14.3$. The histogram is analyzed with a χ^2 goodness-of-fit test for a uniform distribution, where the 95% critical value is 67.5.

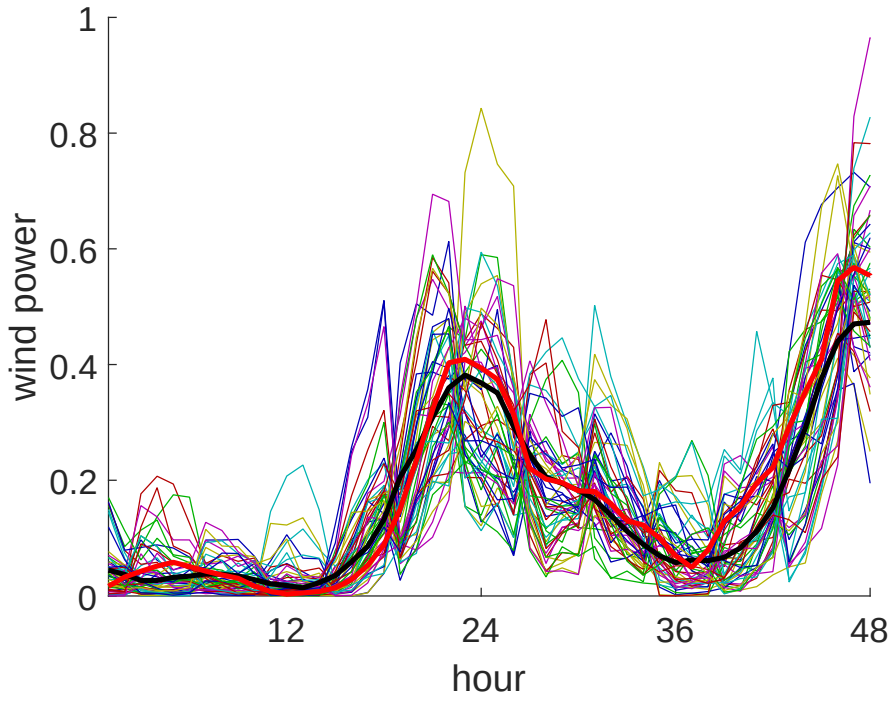


(a) Long Term Generation X1

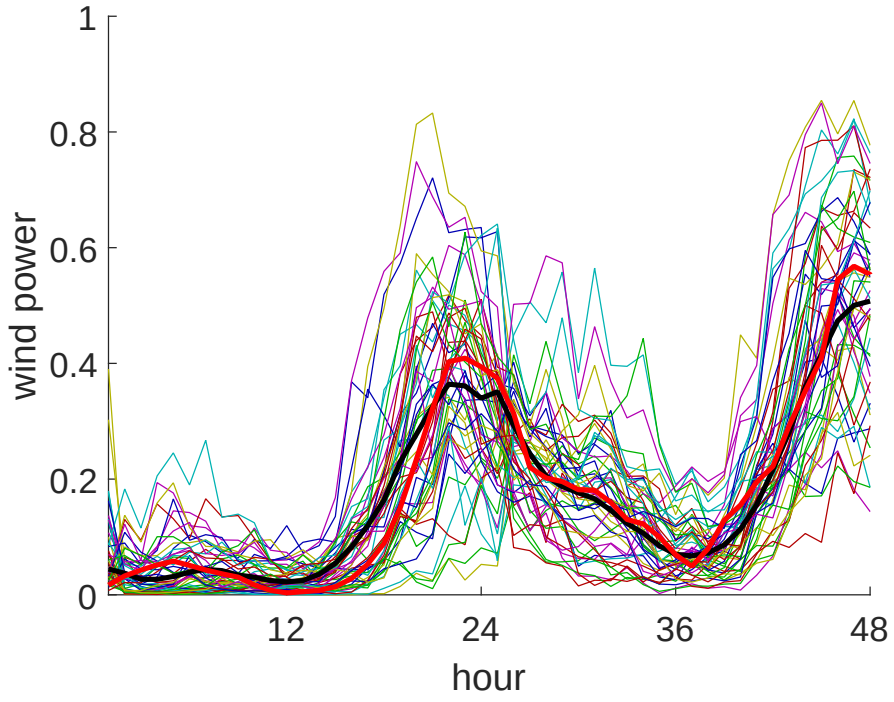


(b) Long Term Generation X24

Figure 3.5: Examples of scenarios generated by two Long Term Generation variations. Hour 1 corresponds to 12:00 on August 5, 2013. The thin lines are wind scenarios. The thick red line is the historical wind outcome. The thick black line is the forecast used to create the marginal distributions.



(a) Naive Concatenation Method NCM4



(b) Naive Concatenation Method NCM24

Figure 3.6: Examples of scenarios generated by two Naive Concatenation Method variations. Hour 1 corresponds to 12:00 on August 5, 2013. The thin lines are wind scenarios. The thick red line is the historical wind outcome. The thick black line is the forecast used to create the marginal distributions.

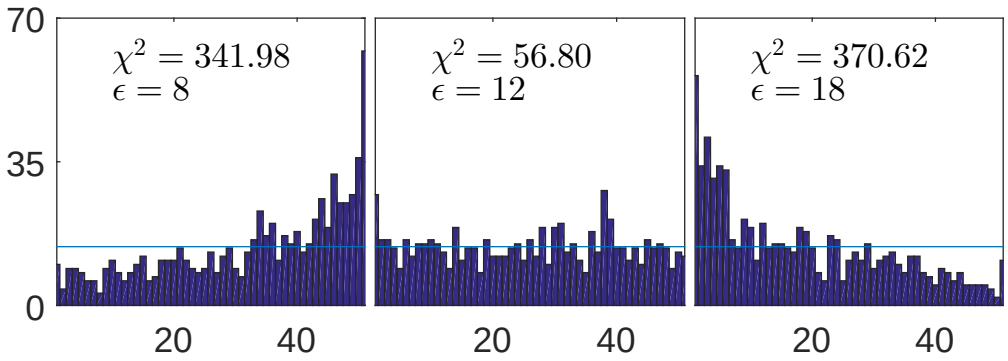
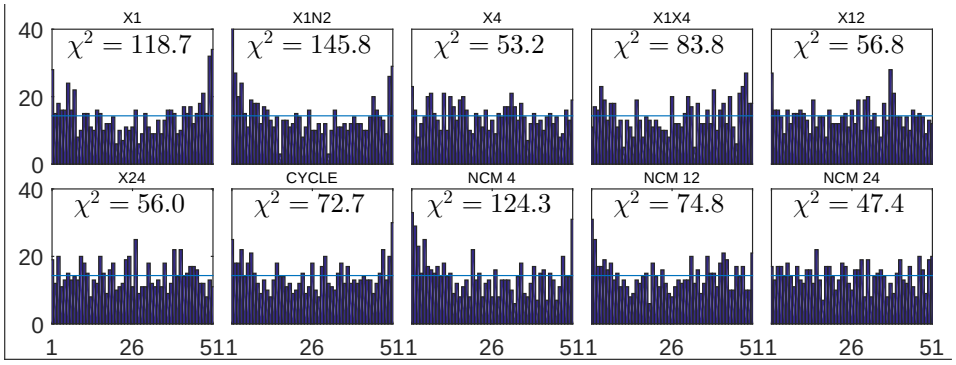
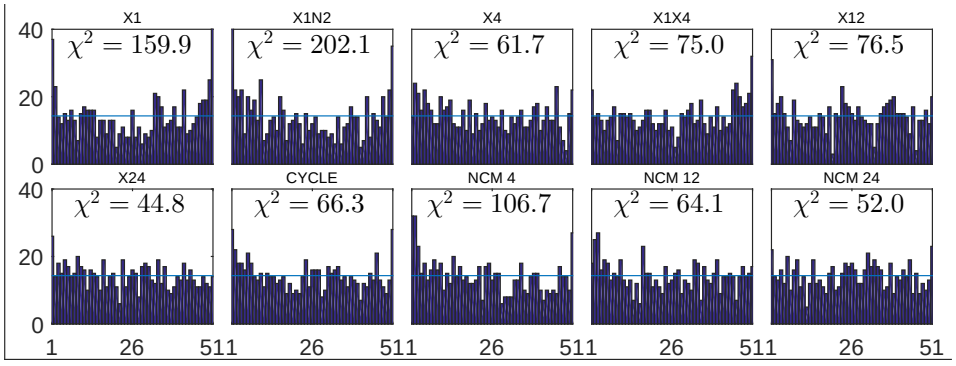


Figure 3.7: Impact of covariance ϵ on MST rank histogram test in variation X12. x-axis is rank (bin), y-axis is count.



(a) Trial 1



(b) Trial 2

Figure 3.8: Minimum Spanning Tree rank histograms. The χ^2 value tests the hypothesis that the histograms are uniform. The 95% χ^2 critical value is 67.5, so values less than 67.5 fail to reject the hypothesis the histogram is uniform.

Figure 3.7 demonstrates the impact of the covariance parameter ϵ. The blue line shows the expected value. When ϵ is too small (left) the scenarios are over

dispersed because the marginal distributions are almost independent, resulting in many high ranks. A large ϵ (right) causes the scenarios to be under dispersed because the marginal distributions are highly correlated, resulting in many low ranks. The covariance parameter for all seven LTG variations and all three NCM variations is therefore selected by optimizing (minimizing) the χ^2 statistic.

Figure 3.8 shows the two results of the MST test for all ten variations, with the χ^2 statistic listed in the histograms. The primary observation is that it is difficult to obtain well-calibrated scenarios when using the previous time step's one time step ahead forecast, F_1^{t-1}, as a predictor variable. This is evidenced by the uniformity test strongly rejecting variations X1, X1N2, and NCM4. The remaining seven variations are all much more well-calibrated.

The Brier Scores are all presented in Table 3.4, and these scores are the average over the two trials. These results tend to show the reverse trends as in the MST test, where X1 and X1X4 perform the best and X4 and X24 perform the worst. This may be because the one time step ahead forecasts are more accurate, allowing the wind scenarios to more closely follow, and therefore have more statistically similar properties to, the historical wind outcome.

Table 3.4: Brier Scores. Bold are best scores, underlined are worst scores.

Event (κ, ξ)	X1	X1N2	X4	X1X4	X12	X24	CYCLE	NCM4	NCM12	NCM24
1. Ramp Down $(1, 0.2)$	**0.002**	**0.002**	0.003	0.003	0.004	0.003	0.003	**0.002**	**0.002**	**0.002**
2. Ramp Up $(1, 0.2)$	0.006	0.006	0.006	**0.005**	0.006	0.006	0.006	0.007	0.006	0.006
3. Ramp Down $(1, 0.4)$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4. Ramp Up $(1, 0.4)$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5. Ramp Down $(3, 0.2)$	**0.024**	0.036	0.038	0.038	0.037	0.037	0.037	0.028	0.029	0.041
6. Ramp Up $(3, 0.2)$	0.027	0.028	0.039	**0.022**	0.038	0.040	0.036	0.036	0.036	0.037
7. Ramp Down $(3, 0.4)$	0.002	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003
8. Ramp Up $(3, 0.4)$	0.007	0.007	0.009	**0.005**	0.009	0.009	0.008	0.009	0.009	0.008
9. Ramp Down $(6, 0.2)$	**0.034**	0.062	0.068	0.053	0.055	0.060	0.063	0.041	0.042	0.065
10. Ramp Up $(6, 0.2)$	0.032	0.033	0.051	**0.027**	0.054	0.058	0.042	0.045	0.049	0.053
11. Ramp Down $(6, 0.4)$	**0.010**	0.019	0.021	0.017	0.017	0.018	0.020	0.012	0.013	0.020
12. Ramp Up $(6, 0.4)$	0.015	0.016	0.024	**0.012**	0.026	0.027	0.020	0.020	0.023	0.025

Finally, the Joint Distribution Comparison results are presented in Figures 3.9, 3.10 and 3.11. The y-axis has been scaled to the 95% critical χ^2 value, so that values less than or equal to 1 indicate that the null hypothesis in Equation (3.10) is not rejected, i.e. the synthetic joint distributions $\hat{P}(F_\delta^{t-L}, W^t)$ are not different from the historical joint distribution $P(F_\delta^{t-L}, W^t)$. The main observation in Figure 3.9, which tests the hypotheses $\mathcal{H}_0(L, \delta = L)$, is that the Long Term Generation variations X1 and X1N2 perform the best over all lead times L, with variation X1X4 having the next best performance.

While NCM4 performs adequately, but still worse than X1, X1N2, and X1X4, the Naive Concatenation Methods perform much worse as $\hat{\Delta}$ is increased. NCM24 is the second-worst algorithm according to the JDC test, with X24 the worst. Finally, observe that all ten variations have difficulty drawing scenarios from the same joint distribution as historical data when L is small, such as when $L \leq 4$.

Figure 3.10 and 3.11 show the JDC results when $\delta > L$ and $\delta < L$, respectively. In all cases the relative performances of the seven LTG and three NCM variations remain the same, with LTG variations X1, X1N2, and X1X4 always performing the best and LTG variation X24 and NCM variation NCM24 performing the worst. Figure 3.10 shows that it becomes easier to obtain similar joint distributions whenever $\delta > L$. Figure 3.11 shows a similar trend. However, these results indicate that when $\delta = L - 1$ and $\delta = L - 2$ it initially becomes more difficult to draw scenarios from the same joint distribution when $L \leq 4$.

Altogether, these results show that the Long Term Generation variations X1X4 produces the most statistically similar long-term wind scenarios as compared to the historical data: the MST test shows the X1X4 variation is almost well-calibrated; the Brier Scores show that X1X4, alongside X1, is the most statistically similar;

and the JDC test demonstrates the synthetic joint distributions are similar to the historical distributions. Without the JDC test the Naive Concatenation Method variations NCM12 or NCM24 may appear suitable since they are well-calibrated and have mediocre Brier Scores. However, the JDC test demonstrates these naive methods draw scenarios from a statistically different joint distribution than historical data.

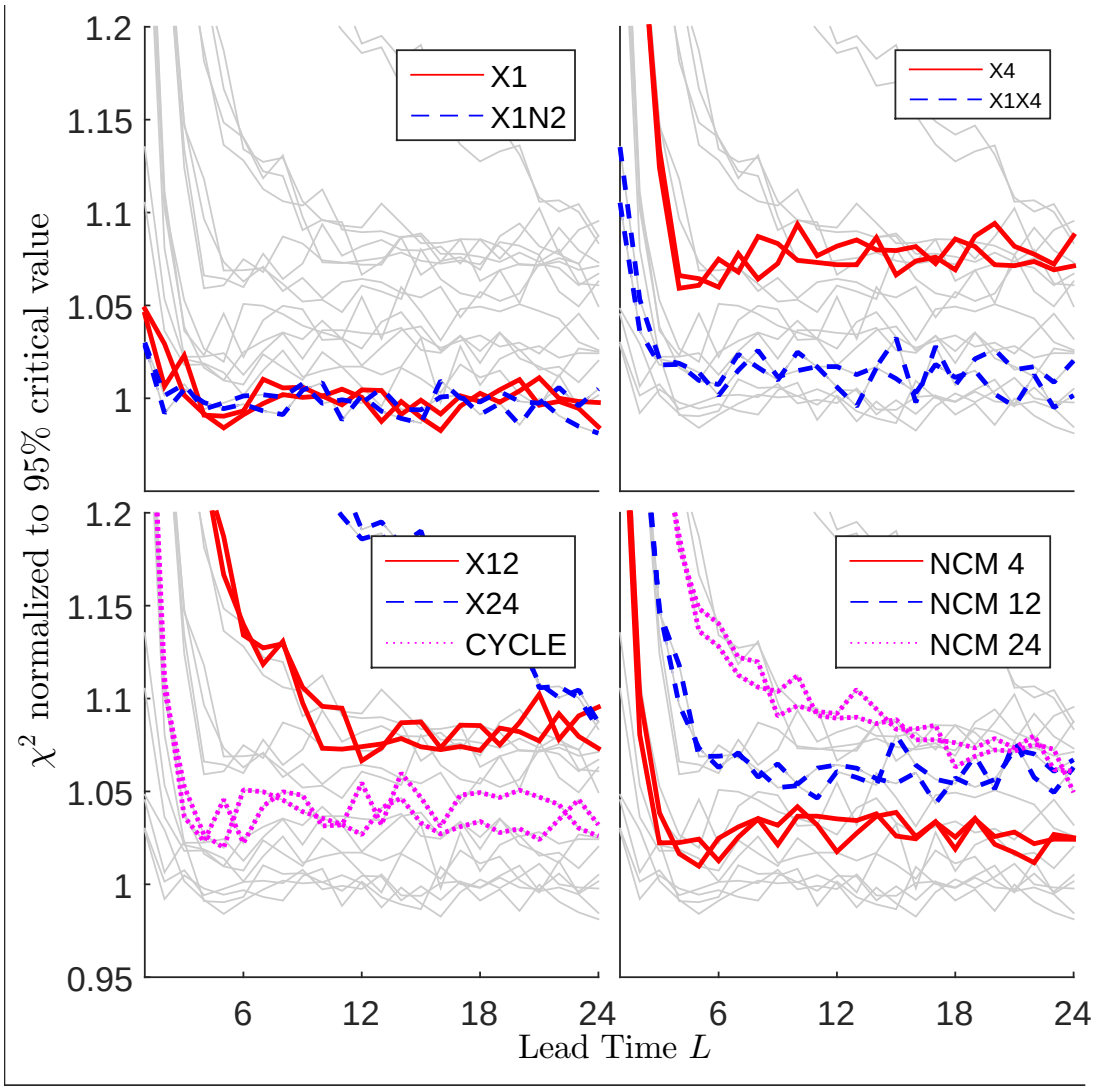
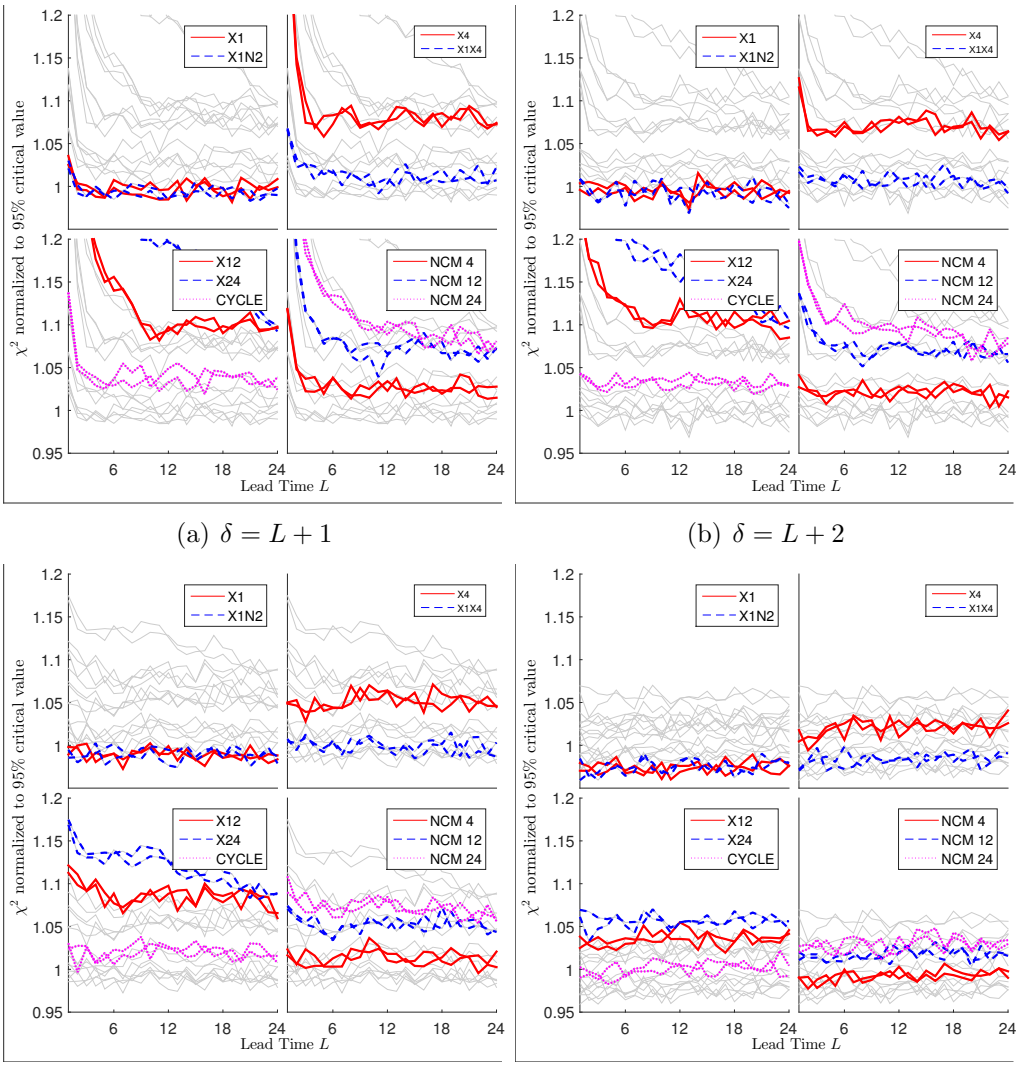


Figure 3.9: Joint Distribution Comparison results, when $\delta = 0$. Small values on the y-axis indicate that the joint distribution resulting from the synthetic wind scenarios is more statistically similar to the historical joint distribution, where 1 is the critical value.

3.8 Note on Optimized Covariance Parameter

An interesting pattern emerged from the Long Term Generation covariance parameters ϵ optimized using the Minimum Spanning Tree rank histogram test. First,



(a) $\delta = L + 1$

(b) $\delta = L + 2$

(c) $\delta = L + 3$

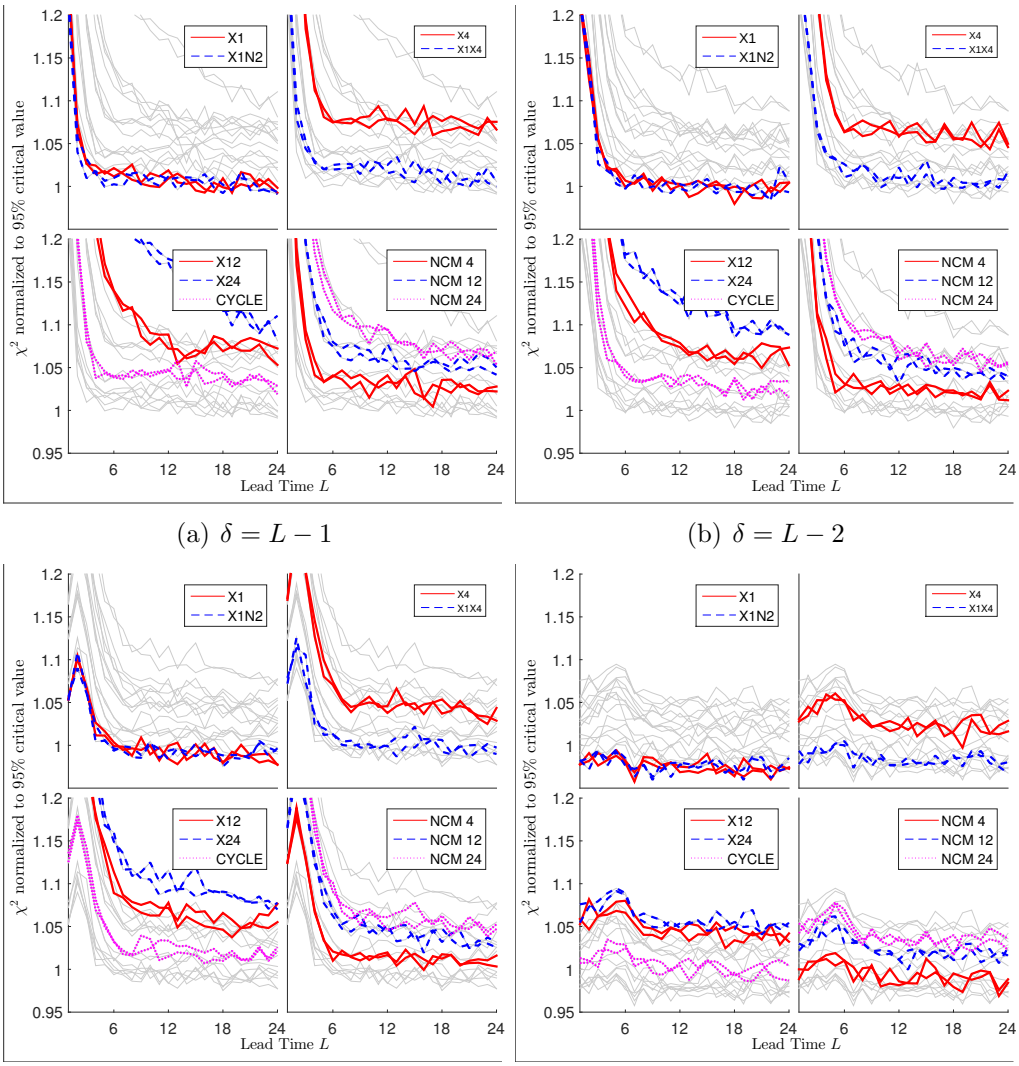
(d) $\delta = L + 6$

Figure 3.10: Joint Distribution Comparison results with $\delta > L$

define

$$\sigma_L^2 = var(\{f_L^{t-L} - w^t\}_t) \tag{3.26}$$

to be the variance of the historical error between the wind outcome and the L time step ahead forecast. Using historical data from August 1, 2013 to August 1, 2014, the variances are computed and shown in Table 3.5. Also, let ϵ_L be the optimized exponential covariance parameter using the L time step ahead predictor variable f_L^{t-L}; that is, $\epsilon_1, \epsilon_4, \epsilon_{12}$, and ϵ_{24} are the covariance parameters for LTG variations



(a) $\delta = L - 1$ (b) $\delta = L - 2$

(c) $\delta = L - 3$ (d) $\delta = L - 6$

Figure 3.11: Joint Distribution Comparison results with $\delta < L$

X1, X4, X12, and X24, respectively . These optimized parameters shown in Table 3.1 are repeated in Table 3.5.

Using the Gaussian copula method, the covariance between the random variables W^i and W^j in the transformed space is $exp(-|i-j|/\epsilon_L)$. If it is assumed that the historical errors $f_L^{t-L} - w^t$ follow a normal distribution with variance σ_L^2, then in the untransformed space the covariance is instead $exp(-|i-j|/\epsilon_L)\sigma_L^2$. Letting $n = |i-j|$, the covariance in the untransformed space is plotted in Figure 3.12.

Table 3.5: Historical variance of errors (first row), optimized exponential covariance parameter (second row), and first derivative of covariance function (third row).

	$L=1$	$L=4$	$L=12$	$L=24$
$\sigma_L^2 \cdot 10^3$	2.7	5.8	7.5	9.1
ϵ_L	4.5	9	12	14
first derivative $\times 10^{-4}$ at $n=0$	6.100	6.431	6.283	6.494

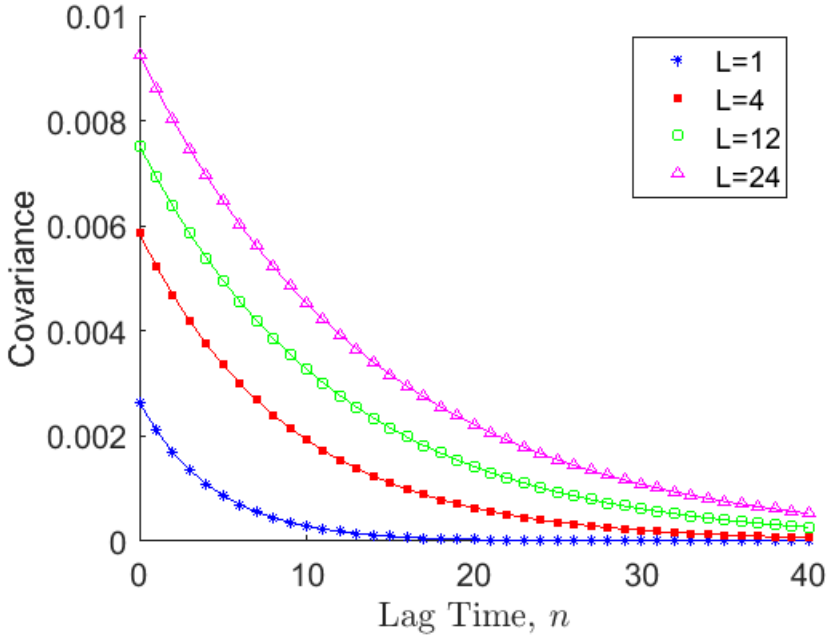


Figure 3.12: Covariance function in untransformed space.

Looking at these curves, the change in covariance from $n=0$ to $n=1$ of all four curves are approximately the same. Letting n be a continuous variable, the derivative at $n=0$ is $-\sigma_L^2/\epsilon_L$. The numerical values of these derivatives are shown in the last column of Table 3.5. The mean value is $-6.32 \cdot 10^{-4}$, and the difference between the largest (from $L=24$) and smallest derivative (from $L=1$) is 12.5%.

This suggests that to obtain well-calibrated scenarios according to the MST rank histogram test it is most important to obtain the correct change in covariance between random variables W^i and W^j when $|i-j|$ is small. If this pattern holds,

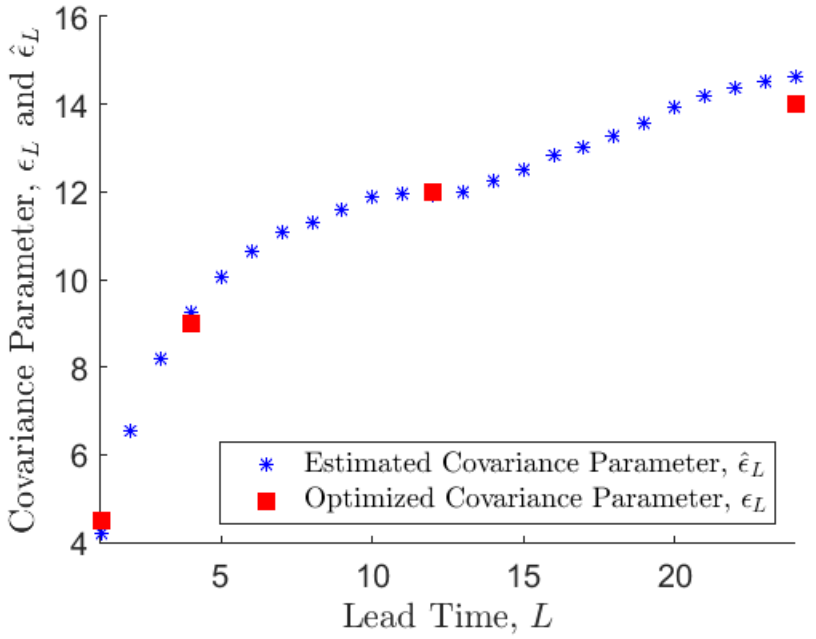


Figure 3.13: Optimized and estimated optimal exponential covariance parameters.

then when using a predictor variable f_L^{t-L} the optimized exponential covariance parameter can be estimated as

$$\hat{\epsilon}_L = \frac{\sigma_L^2}{-6.32 \cdot 10^{-4}}. \tag{3.27}$$

The four optimized covariance parameters and the estimated covariance parameters are shown in Figure 3.13. This pattern of first derivative matching may hold in other wind datasets, but it should not be assumed that the same constant $-6.32 \cdot 10^{-4}$ will appear.

3.9 Conclusions

In this Chapter the new Long Term Generation (LTG) method was developed. The LTG method follows the marginal-then-joint approach of first estimating the

marginal distributions at each time step over a long horizon and then forming the joint distribution. The Minimum Spanning Tree rank histograms and Brier Score showed that scenarios generated by LTG on a data set from Bonneville Power Administration had similar statistical properties as the historical wind outcomes. In particular, the new Joint Distribution Comparison also demonstrated the joint distribution of synthetic scenarios and historical forecasts was similar to the historical joint distribution, unlike scenarios generated by the Naive Concatenation Method. These results all show LTG can generate long-term wind scenarios that can be used to simulate power systems with wind power integration in order to help evaluate their performance.

BIBLIOGRAPHY

[1] Wind Generation & Total Load in The BPA Balancing Authority. `https://transmission.bpa.gov/business/operations/wind/`. Accessed: 2016-10-20.

[2] Wind Generation Capacity in the BPA Balancing Authority Area. `https://transmission.bpa.gov/business/operations/wind/WIND\_InstalledCapacity\_PLOT.pdf`. Accessed: 2016-10-20.

[3] John S Anagnostopoulos and Dimitris E Papantonis. Pumping station design for a pumped-storage wind-hydro power plant. *Energy Conversion and Management*, 48(11):3009–3017, 2007.

[4] Rüdiger Barth, Heike Brand, Peter Meibom, and Christoph Weber. A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on*, pages 1–8. IEEE, 2006.

[5] Rüdiger Barth, Lennart Söder, Christoph Weber, Heike Brand, and Derk Jan Swider. Methodology of the scenario tree tool. *Wilmar Deliverable*, 6, 2006.

[6] Ricardo J Bessa, Joana Mendes, Vladimiro Miranda, Audun Botterud, Jianui Wang, and Zhi Zhou. Quantile-copula density forecast for wind power uncertainty modeling. In *PowerTech, 2011 IEEE Trondheim*, pages 1–8. IEEE, 2011.

[7] Ricardo J Bessa, Vladimiro Miranda, Audun Botterud, Jianhui Wang, and Emil M Constantinescu. Time adaptive conditional kernel density estimation for wind power forecasting. *IEEE Transactions on Sustainable Energy*, 3(4):660–669, 2012.

[8] Hans Bludszuweit, José Antonio Domínguez-Navarro, and Andrés Llombart. Statistical analysis of wind power forecast error. *IEEE Transactions on Power Systems*, 23(3):983–991, 2008.

[9] Taras Bodnar and Nikolaus Hautsch. Copula-based dynamic conditional correlation multiplicative error processes. *Available at SSRN 2144741*, 2012.

[10] John Bjørnar Bremnes. Probabilistic wind power forecasts using local quantile regression. *Wind Energy*, 7(1):47–54, 2004.

[11] John Bjørnar Bremnes. A comparison of a few statistical models for making quantile wind power forecasts. *Wind Energy*, 9(1-2):3–11, 2006.

[12] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[13] Michael Brower et al. Development of eastern regional wind resource and wind plant output datasets. *Rep. No. NREL/SR-550*, 46764, 2009.

[14] Song Xi Chen. Beta kernel estimators for density functions. *Computational Statistics & Data Analysis*, 31(2):131–145, 1999.

[15] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in Statistics*, pages 66–70. Springer, 1992.

[16] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

[17] Norbert Henze. A multivariate two-sample test based on the number of nearest neighbor type coincidences. *The Annals of Statistics*, pages 772–783, 1988.

[18] Cody A Hill, Matthew Clayton Such, Dongmei Chen, Juan Gonzalez, and W Mack Grady. Battery energy storage for enabling integration of distributed solar power generation. *IEEE Transactions on smart grid*, 3(2):850–857, 2012.

[19] MS Jamel, A Abd Rahman, and AH Shamsuddin. Advances in the integration of solar thermal energy with conventional and non-conventional power plants. *Renewable and Sustainable Energy Reviews*, 20:71–81, 2013.

[20] Ruiwei Jiang, Jianhui Wang, and Yongpei Guan. Robust unit commitment with wind power and pumped storage hydro. *IEEE Transactions on Power Systems*, 27(2):800–810, 2012.

[21] Jeremie Juban, Nils Siebert, and George N Kariniotakis. Probabilistic short-term wind power forecasting for the optimal management of wind generation. In *Power Tech, 2007 IEEE Lausanne*, pages 683–688. IEEE, 2007.

[22] Xi-Yuan Ma, Yuan-Zhang Sun, and Hua-Liang Fang. Scenario generation of wind power based on statistical uncertainty and variability. *IEEE Transactions on Sustainable Energy*, 4(4):894–904, 2013.

[23] YANG Ming, LIN You, ZHU Simeng, HAN Xueshan, and WANG Hongtao. Multi-dimensional scenario forecast for generation of multiple wind farms. *Journal of Modern Power Systems and Clean Energy*, 3(3):361–370, 2015.

[24] Jan Kloppenborg Møller, Henrik Aalborg Nielsen, and Henrik Madsen. Time-adaptive quantile regression. *Computational Statistics & Data Analysis*, 52(3):1292–1303, 2008.

[25] Pierre Pinson. Very-short-term probabilistic forecasting of wind power with generalized logit–normal distributions. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(4):555–576, 2012.

[26] Pierre Pinson and Robin Girard. Evaluating the quality of scenarios of short-term wind power generation. *Applied Energy*, 96:12–20, 2012.

[27] Pierre Pinson and George Kariniotakis. Conditional prediction intervals of wind power generation. *IEEE Transactions on Power Systems*, 25(4):1845–1856, 2010.

[28] Pierre Pinson, Henrik Madsen, Henrik Aa Nielsen, George Papaefthymiou, and Bernd Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind energy*, 12(1):51–62, 2009.

[29] Rommel G Regis and Christine A Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

[30] Didem Sari, Youngrok Lee, Sarah Ryan, and David Woodruff. Statistical metrics for assessing the quality of wind power scenarios for stochastic unit commitment. *Wind Energy*, 2015.

[31] Saurabh Tewari, Charles J Geyer, and Ned Mohan. A statistical model for wind power forecast error and its application to the estimation of penalties in liberalized markets. *IEEE Transactions on Power Systems*, 26(4):2031–2039, 2011.

[32] Aidan Tuohy, Eleanor Denny, and Mark O'Malley. Rolling unit commitment for systems with significant installed wind capacity. In *Power Tech, 2007 IEEE Lausanne*, pages 1380–1385. IEEE, 2007.

[33] Aidan Tuohy, Peter Meibom, Eleanor Denny, and Mark O'Malley. Unit com-

mitment for systems with significant wind penetration. *IEEE Transactions on Power Systems*, 24(2):592–601, 2009.

[34] Bart C Ummels, Madeleine Gibescu, Engbert Pelgrum, Wil L Kling, and Arno J Brand. Impacts of wind power on thermal generation unit commitment and dispatch. *IEEE Transactions on energy conversion*, 22(1):44–51, 2007.

[35] J Wang, A Botterud, R Bessa, H Keko, L Carvalho, D Issicaba, J Sumaili, and V Miranda. Wind power forecasting uncertainty and unit commitment. *Applied Energy*, 88(11):4014–4023, 2011.

[36] Daniel S Wilks. The minimum spanning tree histogram as a verification tool for multidimensional ensemble forecasts. *Monthly Weather Review*, 132(6):1329–1340, 2004.

CHAPTER 4

OPERATION OF WIND AND HYDROPOWER PRODUCER WITH MARKET INFLUENCE

4.1 Abstract

In this Chapter we propose a rolling horizon formulation of a wind and hydropower producer with market influence on the day ahead and hourly intraday markets. In the Combined model a single power producer markets both the wind and hydropower, whereas in the Individual model the wind producer and hydropower producer act independently. Treating the power producer as a price setter in both markets results in a nonlinear problem, and, furthermore, it is nonconvex due to the hydropower production curves. Neuro-dynamic programming is used to determine the optimal controls. A simplified version of both the Combined and Individual models yields a technique to easily obtain the approximate amount of power sold on the day and hour ahead markets. A case study first shows that the net amount of power sold on the markets follows similar trends in both the Combined and Individual models. These results also show that the solution to the simplified model is similar to the solution obtained by the dynamic programming formulation. The simplified model is then used to demonstrate that the differences between the Combined and Individual models become more pronounced as the market influence is increased.

4.2 Introduction

Wind power capacity has significantly increased over the past decade, and much work has been devoted to determining how to best integrate this source into the power grid [25]. The two difficulties are that wind is non-dispatchable, meaning it cannot be controlled, and that wind power forecasts have large uncertainties. By combining wind power with a dispatchable energy source, such as thermal power

or hydropower, the uncertainty in the combined power output decreases [24].

Hydropower in particular, either from pumped hydro storage or hydropower reservoirs along a river, is an ideal energy source to balance wind power because it has a quick ramp time and large capacity. Many aspects of wind with hydropower balancing have been investigated. For example, robust optimization algorithms were proposed to prevent worst-case outcomes [11, 2, 22]; pumped-hydro systems were evaluated to determine the hydropower capacity required to adequately balance wind power [4]; an analysis of balancing wind with hydropower in a congested power grid was performed [15]; and a case study examined the costs of using a hydropower reservoir system with ecological constraints to provide a wind-following service [8].

Previous research, including all of the preceding citations, considers the wind and hydropower power producer to be a price taker. This means the price of power is independent of how much power the producer buys or sells. Other examples of modeling the producer as a price taker include [12, 5, 3, 7]. Most existing work that models the producer as a price setter does not consider the case where there is wind power integration. In [9] the market is dominated by hydropower production and the price of power is an endogenous random variable. In another paper a hydropower producer operating in a market with wind penetration is a price taker on the day ahead market but is a price setter on the hourly intraday market [13].

Additionally, most work uses either linear or at least convex models. Linear formulations can be efficiently solved with linear programming, and convex optimal control problems are often solved using Stochastic Dual Dynamic Programming (SDDP) [9]. However, SDDP also suffers from computational costs when the number of state dimensions increases beyond about 10 [19].

In this Chapter we develop two nonconvex models of how to sell wind and hydropower on the day and hour ahead markets when there is market influence. The Combined model has a single wind and hydropower generation company WH-GENCO, and in the Individual model the wind producer W-GENCO and the hydropower producer H-GENCO act independently.

The data for these models and parts of the optimal control dynamic programming formulation were organized and developed in [21]. The wind and hydropower system is based on the Bonneville Power Administration (BPA), which is a large power producer in the Pacific Northwest. Only the Combined model was considered in [21], and not the Individual model. The other primary differences in the dynamic programming formulation and power producer model presented here as compared to [21] are: the hydropower system includes seven reservoirs instead of two; the hydropower generation curves are nonconcave; the wind power forecasts use historical forecasts generated by BPA; the rolling horizon formulation enacts decisions every eight hours instead of every 24; and the dynamic programming formulation includes in the state space the day ahead power committed during previous days.

Nuero-dynamic programming is used to solve the resulting optimal control problem. In particular we use the FUA method developed in Chapter 2. The formulation depends upon the current hour of day. The state space has between 12 and 16 state dimensions, and there are either three or four stages with a horizon of between 58 and 72 hours.

This Chapter is organized as follows. In Section 4.3 the Combined and Individual models are described. Section 4.4 describes the new rolling horizon formulation. Finally, results are presented in Section 4.5.

4.3 Wind and Hydropower Producer Model

The wind and hydropower producer is modeled on the Bonneville Power Administration, which is a US federal agency that markets power produced by wind turbines and by hydropower reservoirs along the Columbia and Snake Rivers. In this model time steps are in hours. For notation a superscript will always denote hour. The hours of day range from 00 to 23, and the hour i of day k can be written $24k + i$.

The Combined model is described here, and the differences with the Individual model are discussed in Section 4.3.5. In the day ahead market, at every day k at hour 07 the wind and hydropower producer WH-GENCO must make a commitment on how much power will be bought or sold at each of the twenty four hours of the following day, numbered $24(k+1), ..., 24(k+1)+23$. Let da^i be the amount of day ahead committed power, measured in MWh, that WH-GENCO sells at i. Also, let th^i be the total hydropower produced over all individual hydropower reservoirs at hour i and let $wind^i$ be the wind power. WH-GENCO must meet the load, $load^i$, within its balancing area. The net power, net^i, that WH-GENCO buys or sells on the hour ahead intraday market is therefore

$$net^i = wind^i + th^i - load^i - da^i. \tag{4.1}$$

Because WH-GENCO generates such a large amount of power it is modeled as a price setter on both the day ahead and hour ahead markets.

The objective is to maximize the total profits. The decisions that need to be made are how much power to commit on the day ahead market, which is made once daily at hour 07, and how much water to release from each of the reservoirs, which in turn determines the hydropower production and impacts the net power

sold on the hour ahead market. The wind power is assumed to have an imperfect forecast, so these optimal decisions are determined by maximizing an expected profit. The other variables affecting the profits are the price of power, load, and water inflow into the reservoirs from an upstream source (which impacts the hydropower generation). These are all assumed to have a perfect forecast, although the presented model can easily be generalized to include uncertain price, load, and inflow forecasts.

The following three subsections describe the hydropower reservoir system, the wind power model, and the price of power in the Combined and Individual models.

4.3.1 Reservoir State Dynamics and Constraints

Three of the hydropower reservoirs operated by BPA are selected for our model, shown in Figure 4.1 as the shaded reservoirs numbered $1, 3$, and 7. These hydropower reservoirs are Grand Coulee, which has a capacity of 6735 MW, Chief Joseph (2607 MW), and McNary (1120 MW). Altogether, the total capacity of these three reservoirs is 10462 MW, which is 53% of the total capacity of 19693 MW of all reservoirs BPA operates. Four extra run-of-river reservoirs are also included in the model, labeled as $2, 4, 5$ and 6 in Figure 4.1. These reservoirs are used to track the amount of water contained in the river.

It is assumed that the reservoir releases and inflows are constant over periods of 8 hours. This is used to simplify the model and reduce the computation, and in future work this assumption can be relaxed.

For notation, a subscript $r = 1, ..., 7$ will always be used to denote a specific reservoir. Let vol_r^i be the volume of water measured in ksfd ($86.4 \cdot 10^6$ cubic feet)

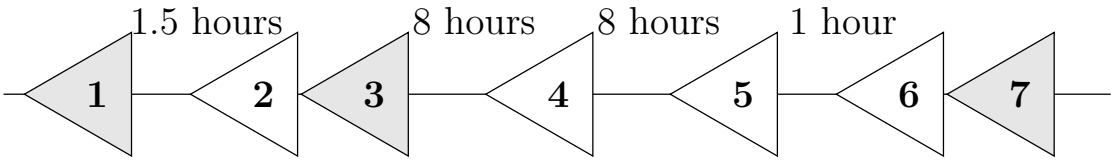


Figure 4.1: Reservoir network diagram, water travels from left to right. Numbers indicate the reservoir number. Gray reservoirs are hydropower reservoirs, with 1, 3, and 7 corresponding to Grand Coulee, Chief Joseph, and McNary, respectively. White reservoirs are run-of-river. The displayed hours indicate how long it takes water to travel from an upstream reservoir to the downstream reservoir.

of reservoir r at hour i. Because the reservoir controls and inflows are in time steps of 8 hours, the power house releases ph_r^i and spillway releases sw_r^i denote the total volume of water released from reservoir r over hours i to $i+7$. The total volume of water released over these 8 hours is the sum

$$rel_r^i = ph_r^i + sw_r^i. \tag{4.2}$$

The run-of-river reservoirs do not have any controls, so

$$rel_r^i = vol_r^i, \quad r = 2, 4, 5, 6. \tag{4.3}$$

Let in_r^i be the total water volume inflow into reservoir r over hours i to $i+7$ from an outside upstream source. Only reservoir 1, Grand Coulee, has inflow from an upstream source, so

$$in_r^i = 0, \quad r = 2, ..., 7. \tag{4.4}$$

Let $vol^i = [vol_1^i, ..., vol_7^i]'$ be the column vector of reservoir water volumes at hour i, and similarly define the vectors ph^i, sw^i, rel^i, and in^i.

The reservoir system state equation is therefore

$$vol^{i+8} = vol^i + in^i + Arel^i, \tag{4.5}$$

where A is the adjacency matrix

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.5/8 & -1 & 0 & 0 & 0 & 0 & 0 \\ 6.5/8 & 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/8 & -1 & 0 \\ 0 & 0 & 0 & 0 & 7/8 & 1 & -1 \end{bmatrix} \tag{4.6}$$

The off-diagonal elements in row i and column $j \neq i$ indicate which portion of the water released from reservoir j during a time step ends up in reservoir i at the next time step. For example, it requires 1.5 hours for water to travel from hydropower reservoir 1, Grand Coulee, to hydropower reservoir 3, Chief Joseph. Because time steps are in 8 hours, the fraction $A_{3,1} = 6.5/8$ of the water released from reservoir 1, rel_1^i, reaches reservoir 3 by the next time step. The remaining fraction $A_{2,1} = 1.5/8$ is still on the river, and so run-of-river reservoir 2 is used to track this information.

There are two types of constraints on the hydropower reservoir power house and spillway releases. The first constraints, listed in Table 4.1, enforce minimum and maximum power house and spillway releases, as well as rate of change of total release. The second constraints, listed in Table 4.2, enforce that during operation the volume of water of each of the three hydropower reservoirs remains between an upper and lower limit. The numbers in parentheses are the absolute reservoir minimum and maximum volumes.

The constraints on maximum total release also place bounds on the water volume in the run-of-river reservoirs, shown in Table 4.3. These bounds are used

reservoir	1 Grand Coulee	3 Chief Joseph	7 McNary
minimum total release	0	0	16.7
minimum power house release	0	0	0
minimum spillway release	0	0	0
maximum power house release	88.4	74.1	78.9
maximum spillway release	334	167	267
maximum change in total release	28.3	12.3	50

Table 4.1: Reservoir release constraints, measured in ksfd/8 hours.

reservoir	1 Grand Coulee	3 Chief Joseph	7 McNary
minimum volume	1500(0)	250(234)	530(482)
maximum volume	1600(2, 6143)	280(299)	560(575)

Table 4.2: Operational hydropower reservoir volume constraints, measured in ksfd. Numbers in parentheses are absolute reservoir bounds.

for sampling the state space when applying the neuro-dynamic programming algorithm. The minimum water volume in all four is 0 because the minimum total release from the upstream hydropower reservoirs (either 1 or 3) is 0. The maximum volume of run-of-river reservoir 2 is calculated by taking the maximum release from the upstream reservoir 1, which is 334, and multiplying it by the fraction of the upstream release that ends up in reservoir 2, which is 1.5/8. This yields $62.6 = 334 \cdot 1.5/8$. The maximum volumes for run-of-river reservoirs $4, 5$ and 6 are calculated in a similar manner.

reservoir	2	4	5	6
minimum volume	0	0	0	0
maximum volume	62.6	167	167	20.8

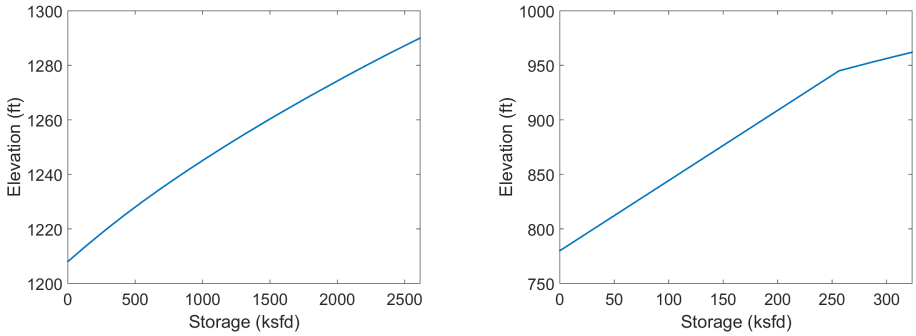
Table 4.3: Run-of-river reservoir volume bounds, measured in ksfd.

4.3.2 Hydropower Generation

The amount of hydropower generated at hydropower reservoir $r = 1, 3, 7$ at hour i is a function of the power house release ph_r^i, the spillway release sw_r^i, and the initial volume of water in the reservoir vol_r^i. First, the water head must be calculated. The elevation of the water, $elev_r(vol_r^i)$, is a nonlinear function of the volume of water, and the tailwater, $tw_r(ph_r^i + sw_r)$, elevation of the river on the downstream side of the reservoir is a function of the total water release. The water head is then

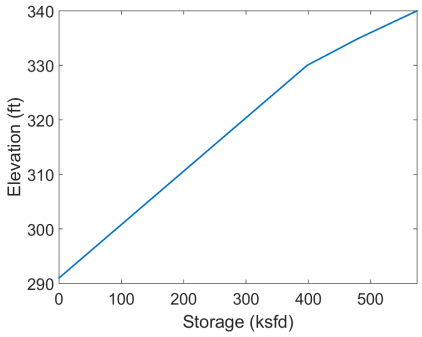
$$head_r^i(vol_r^i, ph_r^i + sw_r^i) = elev_r(vol_r^i) - tw_r(ph_r^i + sw_r^i). \tag{4.7}$$

The functions $elev_r$ and tw_r are plotted in Figures 4.2 and 4.3.



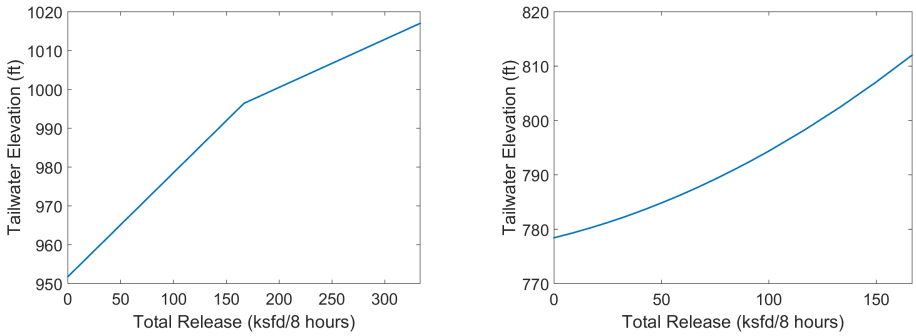
(a) Grand Coulee

(b) Chief Joseph



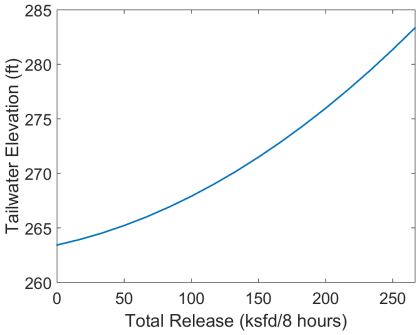
(c) McNary

Figure 4.2: Plotting the function $elev_r, r = 1, 3, 7$, showing the water level elevation as a function of reservoir storage volume.



(a) Grand Coulee

(b) Chief Joseph



(c) McNary

Figure 4.3: Plotting the function $tw_r, r = 1, 3, 7$, showing the tailwater elevation as a function of total reservoir release.

The hydropower generation $hydro_r^i$ at reservoir r and hour i,

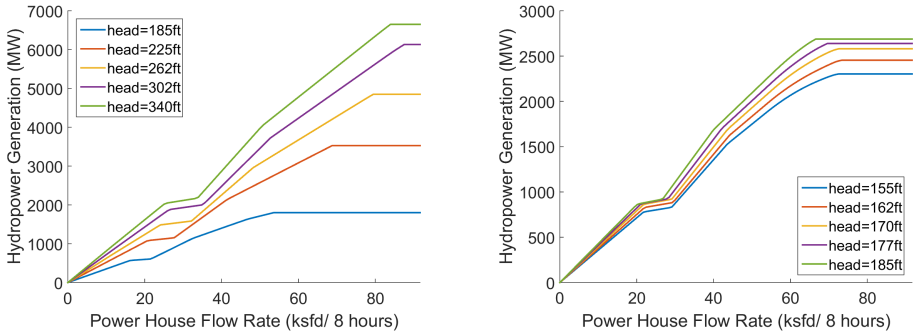
$$hydro_r^i(vol_r^i, ph_r^i, sw_r^i) = gen_r(head_r(vol_r^i, ph_r^i + sw_r^i), ph_r^i), \tag{4.8}$$

is a function of the head and power house release. These generation curves are plotted in Figure 4.4, and use data from [21]. These power generation curves show that for each head there is some maximum power generation. Beyond this point any additional power house release is instead diverted through the spillway, which is shown by the curves reaching a maximum value and then remaining constant. The total hydropower generation th^i at hour i is the sum

$$th^i = \sum_{r=1,3,7} hydro_r^i(vol_r^i, ph_r^i, sw_r^i). \tag{4.9}$$

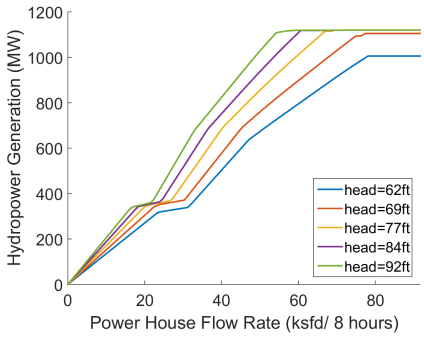
The generation curves are not concave. The large bumps in the middle were manually added in order to demonstrate that neuro-dynamic programming is a

general algorithm that can be applied to nonconvex problems. In reality, power generation curves are often nonconcave, and most models smooth them out.



(a) Grand Coulee

(b) Chief Joseph



(c) McNary

Figure 4.4: Plotting the function $gen_r, r = 1, 3, 7$, showing the hydropower generation as a function of power house release at five head levels.

4.3.3 Wind Power Forecasts and Scenarios

At every hour i BPA creates an hourly wind power forecast that extends 72 hours ahead. Historical forecasts and the actual wind power generation are publicly available at [1]. The historical wind data is scaled so that the wind capacity is 25% of the 10462MW hydropower capacity. For comparison, as of 2013 the capacity of wind power that BPA markets was approximately 22% of the hydropower capacity.

The forecasts are point forecasts, meaning that if the forecast was generated at

hour i then for each $k = 1, ..., 72$ the wind power forecast at hour $i + k$ is a scalar value. However, research has shown the benefit of considering the distribution over possible wind power outcomes as opposed to the single point forecast [23, 18].

Therefore, following the methods in [17, 14, 16], which were described in Chapter 3, historical data is used to first estimate a probabilistic forecast conditioned on the point forecast. Then, equally-likely scenarios can be drawn from this distribution. An example is shown in Figure 4.5. The thick black line is the wind power point forecast that was generated at time 0, the thick red line is the actual wind power generation, and all of the thin lines are scenarios that were generated conditioned on the point forecast. Hour 1 corresponds to hour 08 of August 1, 2013.

In the development of the rolling horizon control formulation it is necessary to obtain the probability of transitioning between scenarios from hour i to hour $i+1$. To generate synthetic scenarios, all three references [17, 14, 16] create an estimate of the joint distribution $\hat{P}^{i,i+1}_{wind}(W^i = wind^i, W^{i+1} = wind^{i+1})$ of the wind scenario random variables W^i and W^{i+1} obtaining values $wind^i$ and $wind^{i+1}$ at hours i and $i+1$, respectively. If N_{WS} wind scenarios were drawn, this joint distribution can be used to calculate the probability of transitioning from a wind scenario with value $wind^i$ at hour i to wind scenario number $1 \leq n \leq N_{WS}$ with value $w^{i+1}(n)$ as

$$P^{i,i+1}_{wind}(wind^i, n) = \frac{\hat{P}^{i,i+1}_{wind}\big(wind^i, w^{i+1}(n)\big)}{\sum_{m=1}^{N_{WS}} \hat{P}^{i,i+1}_{wind}\big(wind^i, w^{i+1}(m)\big)}. \tag{4.10}$$

Additionally, long-term wind scenarios are generated conditioned on the forecasts using the Long Term Generation method developed in Chapter 3. Figure 4.6 shows examples, where the thick black line is the one-hour-ahead forecast (which is different from the forecast in Figure 4.5), the thick red line is the historical wind power outcome, and the thin lines are wind outcome scenarios. Again, hour 1 is

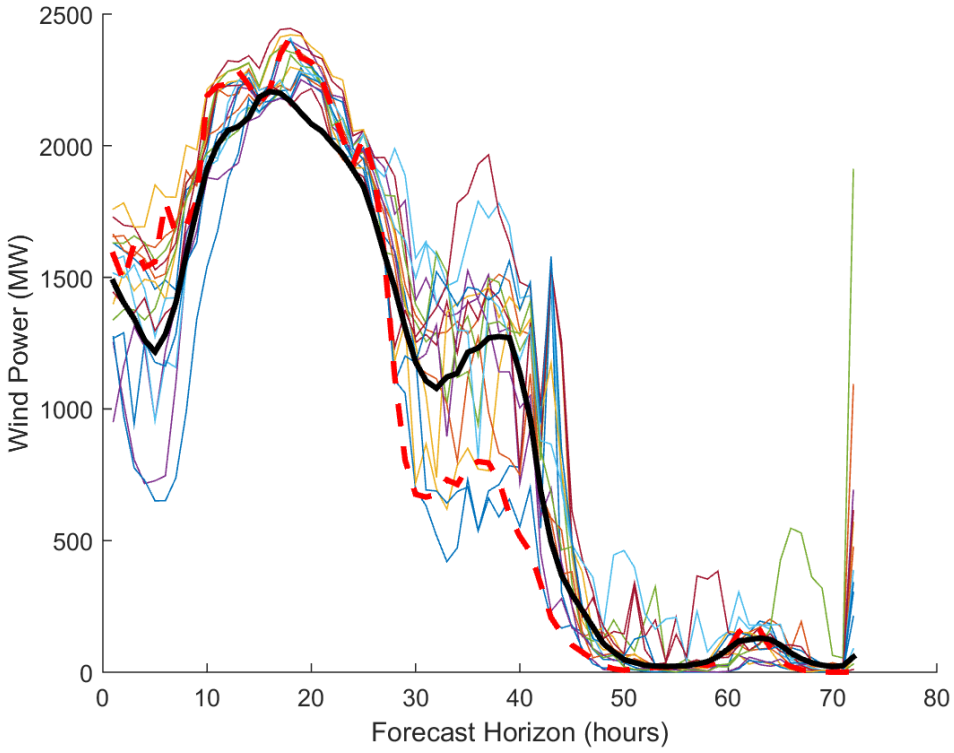


Figure 4.5: Short term wind scenarios conditioned on a single point forecast. The solid black line is the point forecast, the dashed red line is the actual wind outcome, and the thin lines are equally-likely scenarios.

hour 08 of August 1, 2013. These scenarios are used to evaluate the performance of the developed control model.

4.3.4 Combined Model Price Functions

In the Combined model a single power producer WH-GENCO markets both the wind power and the hydropower, and it also determines the reservoir releases. A piecewise linear model is used for the price of power on the day ahead and hour ahead markets, and the price is assumed to have a perfect forecast. The data for this model is from [21]. Given that WH-GENCO committed da^i MWh of power

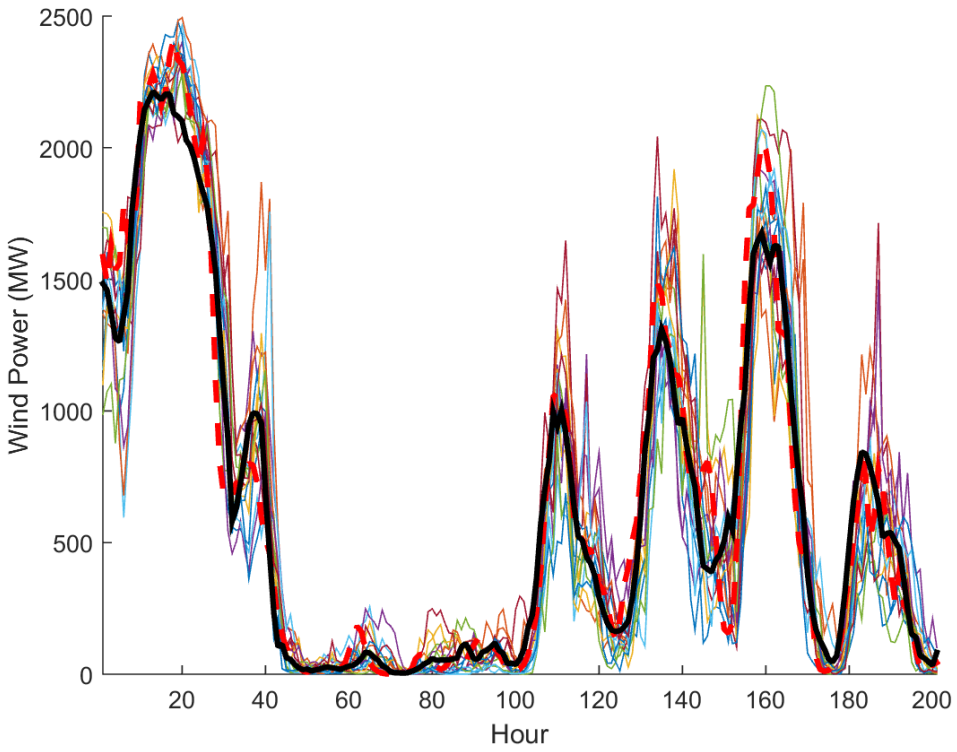


Figure 4.6: Long term wind scenarios. The solid black line is the point forecast, the dashed red line is the actual wind outcome, and the thin lines are equally-likely scenarios.

on the day ahead market, the price of day ahead power is

$$price^i_{DA}(da^i) = base^i_{DA} - da^i \frac{base^i_{DA}}{depth^i_{DA}}. \tag{4.11}$$

Here, $base^i_{DA}$ and $depth^i_{DA}$ are the day ahead base price power and the market depth, respectively. If WH-GENCO commits more than $depth^i_{DA}$ then the price of power becomes negative, reflecting the situation where other customers must be paid to take power. The price of power on the hour ahead intraday market is the piecewise linear function

$$price^i_{HA}(net^i) = \begin{cases} base^i_{HA} - net^i \frac{base^i_{HA}}{depth^i_{HA}} & net^i \geq 0 \\ 1.1base^i_{DA} - net^i \frac{1.1base^i_{DA}}{depth^i_{HA}} & net^i < 0 \end{cases} \tag{4.12}$$

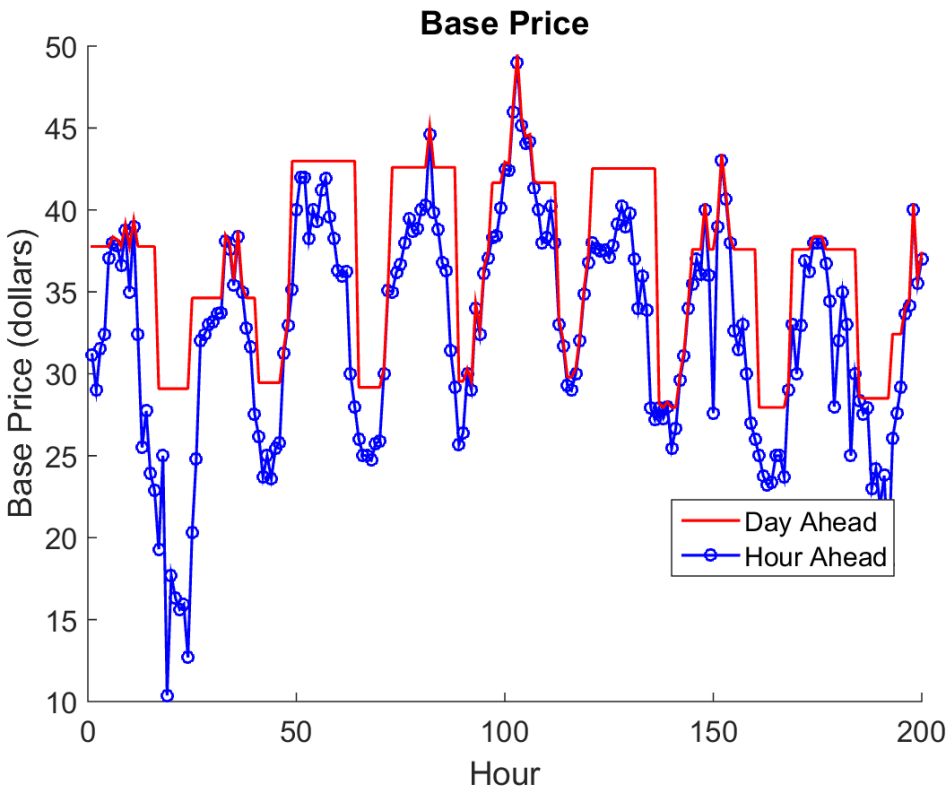


Figure 4.7: Day ahead and Hour Ahead base prices, $base^i_{DA}$ and $base^i_{HA}$. Hour $i = 1$ is hour 08 of August 1, 2013.

The profit at hour i is the product of power and price, so that

$$profit^i_{DA}(da^i) = da^i \cdot \left(base^i_{DA} - da^i \frac{base^i_{DA}}{depth^i_{DA}}\right) \tag{4.13}$$

and

$$profit^i_{HA}(net^i) = net^i \cdot price^i_{HA}(net^i). \tag{4.14}$$

The day ahead and hour ahead base prices are shown in Figure 4.7. The day ahead base price is always greater than the Hour Ahead base price. The day ahead market depth depends on whether the hour is on-peak, between hours 08 and 23 of any day, or off-peak, between hours 00 and 07. The on-peak day ahead market depth is 3000MW and the off-peak depth is 1500MW. The Hour Ahead market depth is always 1000MW. This price model prevents arbitrage between the two markets,

meaning a profit loss will always be incurred for buying power and then selling an equal amount of power in either market at any hour.

4.3.5 Individual Model Price Functions

In the Individual model a wind producer, W-GENCO, and hydropower producer, H-GENCO, sell power independently. W-GENCO acts without considering how the actions of H-GENCO will impact the price of power on the power market. This is a reasonable assumption when the market influence is small, meaning the price of power is almost independent of the actions taken by W-GENCO and H-GENCO. This is also a reasonable assumption under moderate market influence because W-GENCO may not know what constraints or objective function H-GENCO is using to make decisions, and so W-GENCO does not readily have a way to model the actions take by H-GENCO. If the market influence is large then this assumption is still a good first approximation, but in future work different information models can be investigated. For example, in the large market influence case W-GENCO could create and use a simplified model of H-GENCO's behavior.

At day $k-1$ and hour 07, W-GENCO therefore determines how much power to commit on the day ahead market by solving the stochastic optimization problem

$$\max_{\substack{da_W^i \\ i=24k,\ldots,25k-1}} \mathop{\mathbb{E}}_{wind^i} \sum_{i=24k}^{25k-1} price_{DA}^i(da_W^i)da_W^i + price_{HA}^i(wind^i - da_W^i)(wind^i - da_W^i). \tag{4.15}$$

The day and hour ahead prices are the same as in Equations (4.11) and (4.12), and the amount of power sold on the hour ahead market is $wind^i - da_W^i$. The expectation is computed over wind scenarios that are conditioned on the wind forecast generated at hour 07 of day $k-1$.

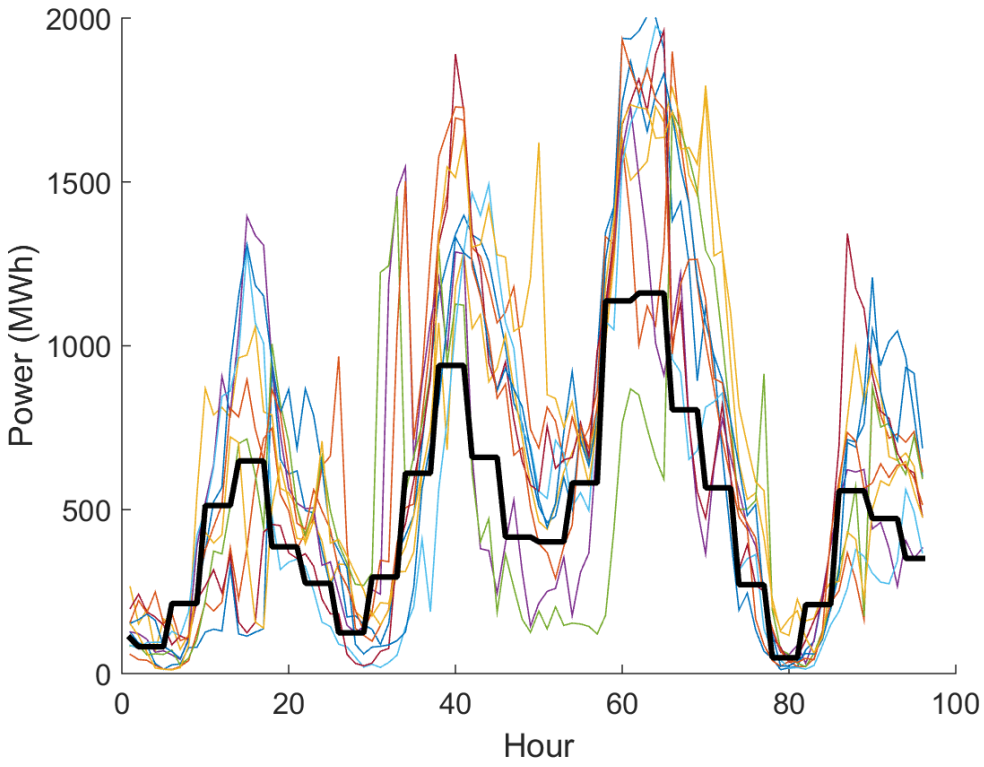


Figure 4.8: The thick black line is the day ahead power committed by W-GENCO in the Individual model, and the thin lines show examples of the wind power forecast scenarios. Hour 1 is August 5, 2013 at hour 08.

This optimization problem is separable, so that the amount of power committed on the day ahead market at hour i is only a function of the forecasted distribution of wind power that will occur at hour i. The black line in Figure 4.8 shows the power sold by W-GENCO on the day ahead market, and the thinner lines are examples of scenarios generated from the wind power forecast.

H-GENCO takes into account the actions of W-GENCO when deciding how much power to sell on the day ahead market and the reservoir releases. This is reasonable because it can be assumed that both W-GENCO and H-GENCO know that W-GENCO is making decisions so as to maximize profits. Additionally, H-GENCO must still supply the load demand $load^i$. Therefore, let $(da_W^i)^*$ be

the amount of power W-GENCO commits at hour i, which both W-GENCO and H-GENCO can calculate. The day ahead price function H-GENCO uses is

$$price^i_{DA,I}(da^i_H) = base^i_{DA} - ((da^i_W)^* + da^i_H)\frac{base^i_{DA}}{depth^i_{DA}}. \tag{4.16}$$

Also, letting

$$net^i_W = wind^i - (da^i_W)^* \tag{4.17}$$

and

$$net^i_H = th^i - da^i_H - load^i \tag{4.18}$$

be the amount of power sold on the hour ahead market by W-GENCO and H-GENCO, respectively, H-GENCO uses

$$price^i_{HA,I}(net^i_H) = price^i_{HA}(net^i_H + net^i_W) \tag{4.19}$$

as the price of power on the hour ahead market. H-GENCO determines the amount of power to sell on the day ahead market and how to control the reservoirs using the same dynamic programming formulation as WH-GENCO, except it uses these modified price function.

4.4 Rolling Horizon Control Formulation

The rolling horizon formulation of this control problem is shown in Figure 4.9. The planning horizon extends either $56, 64$, or 72 hours, and decisions are made and enacted once every 8 hours. The first stage is always 8 hours in duration, and the last two stages are always 24 hours and begin at hour 07 of the corresponding day. This setup may require an intermediate second stage of duration 8 or 16 hours, as shown in subfigures (a) and (b).

Decisions: $da^{24(1)+i}$, $i = 0, ..., 23$ ph_r^i and sw_r^i $i = 8, ..., 15$ $r = 1, 3, 7$	Decisions: ph_r^i and sw_r^i $i = 16, ..., 24 + 7$ $r = 1, 3, 7$	Decisions: $da^{24(2)+i}, i = 0, ..., 23$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	Decisions: $da^{24(3)+i}, i = 0, ..., 6$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	time →
hour = 7 current hour	hour = 15	hour = 24(1) + 7	hour = 24(2) + 7	

(a) Current hour of day is 07

Decisions: ph_r^i and sw_r^i $i = 16, ..., 23$ $r = 1, 3, 7$	Decisions: ph_r^i and sw_r^i $i = 24, ..., 24 + 6$ $r = 1, 3, 7$	Decisions: $da^{24(2)+i}, i = 0, ..., 23$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	Decisions: $da^{24(3)+i}, i = 0, ..., 6$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	time →
hour = 15 current hour	hour = 23	hour = 24(1) + 7	hour = 24(2) + 7	

(b) Current hour of day is 15.

Decisions: ph_r^i and sw_r^i $i = 24, ..., 24 + 6$ $r = 1, 3, 7$	Decisions: $da^{24(2)+i}, i = 0, ..., 23$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	Decisions: $da^{24(3)+i}, i = 0, ..., 6$ $ph_r^{24(2)+i}$ and $sw_r^{24(2)+i}$ $i = 8, ..., 24 + 7$ $r = 1, 3, 7$	time →
hour = 23 current hour	hour = 24(1) + 7	hour = 24(2) + 7	

(c) Current hour of day is 23.

Figure 4.9: Rolling horizon model. Vertical lines mark the individual stages, and thick vertical lines denote that the hour of day is 07. The decisions to be made at each stage are listed, and in the rolling horizon model only the first stage decisions are enacted.

4.4.1 Decision Variables

In each stage the decisions always include the power house and spillway releases from the three hydropower reservoirs over the duration of the stage. The number of release decisions is $6 \cdot (\text{stage length})/8$ because there are two decisions for each of the three hydropower reservoirs and these decisions are made once every 8 hours. The release constraints were described in Section 4.3.2.

The day ahead power commitment decisions of how much power to commit over the 24 hours of the next day are only made if the stage begins at hour 07 of some day. It is assumed that the day ahead commitment is constant over a period of 4 hours, and so there are $6 = 24/4$ day ahead power commitment decision variables. Figure 4.9 shows that the exception is the last stage, which only commits day ahead power for the first 8 hours of the next day. This is because the terminal value function is only a function of the reservoir volumes, as explained in Section 4.A.

4.4.2 State Space

The state space depends on the hour of day at which the stage begins, and it consists of three components. Suppose the stage begins at hour i of day k, or equivalently hour $24k + i$. The first component

$$s_{DA}^{24k+i} = \begin{cases} [da^{24k+8}, ..., da^{24k+23}] & \text{if } i = 07 \\ [da^{24k+16}, ..., da^{24(k+1)+23}] & \text{if } i = 15 \\ [da^{24(k+1)}, ..., da^{24(k+1)+23}] & \text{if } i = 23 \end{cases} \tag{4.20}$$

is all of the day ahead power commitments that have previously been made and for which the commitment is still yet to be fulfilled. For example, if the stage begins at hour 07 then commitments have already been made that must be fulfilled over hours 08 to 23 of the current day. Because the day ahead power is committed in blocks of 4 hours, the number of day ahead state space dimensions is equal to the number of hours listed in Equation (4.20) divided by 4. Specifically, there are $4, 8,$ and 6 day ahead state space dimensions if the stage begins at hour $07, 15$ and 23, respectively.

The second state component

$$s_{res}^{24k+i} = [vol_1^{24k+i}, vol_2^{24k+i}, ..., vol_7^{24k+i}] \tag{4.21}$$

is the volume of water in all 7 hydropower and run-of-river reservoirs at the beginning of the stage. The final component

$$s_w^{24k+i} = [wind^{24k+i-1}] \tag{4.22}$$

is the wind power at the previous hour, $i-1$, in order to incorporate the fact that wind is highly correlated. The full state is

$$s^{24k+i} = [s_{DA}^{24k+i}, s_{res}^{24k+i}, s_w^{24k+1}]. \tag{4.23}$$

Altogether, there are $12, 16$, and 14 state space dimensions if the stage begins at hour 07, 15, and 23, respectively.

The wind state is not included in the state space of the first stage. The wind state is used to calculate the probability of the wind scenarios over the current stage. The wind scenarios are generated at the beginning of the first stage conditioned on the most recent wind power forecast, so for the first stage all scenarios are equally likely.

4.4.3 State Transition

The day ahead state at the beginning of the next stage is created by including the day ahead power commitment decisions that were made at the beginning of the previous stage (if the previous stage began at hour 07) and removing the power commitments that were fulfilled over the previous stage. This results in the day ahead state satisfying Equation (4.20).

The reservoir volume state transition is given by Equation (4.5). The wind state transition in Equation (4.10) is stochastic and depends upon the wind power outcome at the end of the previous stage.

4.4.4 Benefit Function

At each stage the benefit function is the profits accrued over the duration of the stage. If a stage begins at hour 07 of day k then the day ahead power commitment is made and these profits are immediately accrued. The day ahead benefit function is

$$B_{DA}^{24k+7}([da^{24(k+1)}, ..., da^{24(k+1)+23}]) = \sum_{i=0}^{23} profit_{DA}^{24(k+1)+i}(da^i). \tag{4.24}$$

All stages, regardless of which hour of the day they begin, include the hour ahead market profits. If a stage begins at hour i of day k and extends H hours then the hour ahead benefit is

$$\begin{aligned} B_{HA}^{24k+i,24k+i+H}(vol^{24k+i}, ph^{24k+i+j}, sw^{24k+i+j}, da^{24k+i+j}, j = 0, ..., H) = \\ \sum_{k=i}^{j} profit_{HA}^{k}(net^i), \end{aligned} \tag{4.25}$$

where net^i is defined in Equation (4.1). The power house and spillway releases $ph^{24k+i+j}, sw^{24k+i+j}, j = 0, ..., H$ are decisions that are made at the beginning of the stage. However, $da^{24k+i+j}, j = 0, ..., H$ may consist of power commitments that were made at a previous stage (and this information is stored in the state) and power commitments that were made at the beginning of the current stage.

If the stage begins at hour 07 of day k and has length H hours then the benefit

function is

$$B^{24k+7,24k+7+H} = B_{DA}^{24k+7} + B_{HA}^{24k+7,24k+7+H}, \tag{4.26}$$

where the function arguments are omitted to keep the notation clear. Otherwise, if a stage begins at day k and hour $i \neq 07$ then

$$B^{24k+i,24k+i+H} = B_{HA}^{24k+7,24k+7+H}. \tag{4.27}$$

4.4.5 Dynamic Programming Equation

If a stage begins at hour 07 of day k and has a duration of H hours then the dynamic programming equation used to calculate the value function V^{24k+7} of state s^{24k+7} is

$$V^{24k+7}(s^{24k+7}) = \max_{\substack{da^{24(k+1)+i},\\ i=0,\ldots,23}} \max_{\substack{ph_r^{24k+i}\\ sw_r^{24k+i}\\ i=8,\ldots,8+H\\ r=1,2,3}} \mathop{\mathbb{E}}_{\substack{wind^i\\ i=8,\ldots,8+H}} \Big[B^{24k+7,24k+7+H} + V^{24k+7+H}(s^{24k+7+H})\Big]. \tag{4.28}$$

Again, the function arguments are omitted. If the stage is the last stage in the rolling horizon model then the day ahead power commitment is maximized over only the first 8 hours of the next day, as opposed to all 24.

If a stage begins at hour $i \neq 07$ of some day k then the dynamic programming equation is

$$V^{24k+i}(s^{24k+i}) = \max_{\substack{ph_r^{24k+i+j}\\ sw_r^{24k+i+j}\\ j=1,\ldots,8+H\\ r=1,2,3}} \mathop{\mathbb{E}}_{\substack{wind^{24k+i+j}\\ j=1,\ldots,8+H}} \Big[B^{24k+i,24k+i+H} + V^{24k+i+H}(s^{24k+i+H})\Big], \tag{4.29}$$

because only the reservoir release decisions are made and not the day ahead power commitment decisions.

In both Equations (4.28) and (4.29) the reservoir releases are constrained to satisfy the volume constraints in Tables 4.2 and 4.3 over all remaining stages in the dynamic programming model. For example, the releases in the first stage are constrained so that there exists a set of releases resulting in reservoir volumes satisfying the constraints in Tables 4.2 and 4.3 over the next four stages. The set of releases satisfying these constraints depend upon the inflow. The set of feasible releases has linear constraints and can be easily calculated because the inflows over all remaining stages is assumed to have a perfect forecast. However, this approach can be extended to the case where the inflows are uncertain using, for example, chance constraints [10].

4.4.6 Neuro-Dynamic Programming Solution

The presented formulation has up to four stages, between 12 and 16 state dimensions, 24 decision variables, and is stochastic. In Chapter 2 a more computationally efficient neuro-dynamic programming algorithm called FUA was developed. FUA is used here to solve the dynamic programming problem. The state space is sampled using a quasi random Sobol sequence [6].

4.4.7 Approximate Optimal Market Solution

Consider a simplified model where the hydropower reservoir operation constraints described in Section 4.3.2 are removed. Instead, over a horizon j to $j+H$ suppose

the only constraints are that the total hydropower production cannot exceed E MWh, where E is some fixed value, and hydropower production must be non-negative. In the Combined simplified model, WH-GENCO only needs to decide how to allocate this power in order to maximize profits. That is, WH-GENCO solves the optimization problem

$$\begin{aligned} \max_{\substack{da^i, th^i \\ i=j,\ldots,j+H}} \; & \mathop{\mathbb{E}}_{wind^i} \sum_{i=j}^{j+H} price^i_{DA}(da^i)da^i + price^i_{HA}(net^i)net^i \\ \text{s.t.} \; & \sum_{i=j}^{j+H} th^i \leq E, th^i \geq 0 \\ & net^i = th^i + wind^i - load^i - da^i \end{aligned} \tag{4.30}$$

The day ahead power da^i and hydropower production th^i are also constrained to be constant over periods of 4 and 8 hours, respectively, in order to match the full dynamic programming formulation. In the analogous Individual model, H-GENCO would use price functions $price^i_{DA,I}$ and $price^i_{HA,I}$, and $net^i = th^i - load^i - da^i_H$.

This relaxation can be used to easily obtain an approximation of the power sold on the day and hour ahead markets in both the Combined and Individual models. In this process, at hour 07 of day k the amount of power to sell on both markets over hours $j = 24(k+1)$ to $j + H = 24(k+1) + 23$ is the solution the optimization problem in Equation (4.30). The expectation is calculated using the wind scenarios generated at hour 07 of day k.

The value E needs to be chosen. The method used here is to fix E as the mean daily hydropower production. Other possible methods include varying E based upon the wind forecast, or calculating the total hydropower production if a certain volume of water were to be released from each of the hydropower reservoirs.

At each hour i this approach yields a single value of the optimal amount of

power to sell on the day ahead market. This is because the optimization problem is independent of the reservoir, wind, and day ahead state. However, at each hour i the distribution of wind power, $wind^i$, yields a distribution of power sold on the hour ahead market, net^i.

The primary advantages of this simplified model are that it is computationally cheap and that, as shown in Section 4.5, it closely matches the optimal solution obtained by the dynamic programming formulation. This model therefore provides a computationally cheap approach for simulating the power sold on the markets over a long term time horizon.

4.5 Results

The results presented here all use reoptimization. In reoptimization a set of N initial states $s(j), j = 1, ..., N$ are randomly selected; for each initial state a sequence of stochastic outcomes, which in this case is the wind power, is generated over all stages; and for each initial state the decisions are calculated at each stage using the policy as determined by neuro-dynamic programming, and the states and costs are updated using the generated wind outcome. Let the resulting profit of the trajectory beginning at state $s(j)$, following the policies over the entire horizon, and observing the associated stochastic outcome be $pr(j)$. Then, as described in Chapter 2, the Mean Expected Profit (MEP) is

$$MEP = \frac{1}{N} \sum_{j=1}^{N} pr(j). \tag{4.31}$$

Here, a larger MEP is better, whereas in Chapter 2 a smaller Mean Expected Cost (MEC) was better. Reoptimization can also be used to analyze the operational controls and the state space trajectories.

The wind scenarios are all generated using the Long Term Generation method developed in Chapter 3. Recall that these scenarios are generated from the distribution that is conditioned on all of the wind power forecasts.

In the Individual model the reoptimization trajectories of W-GENCO and H-GENCO are paired. A single long-term wind scenario is required to compute the price of power on the hour ahead market because the price depends upon the net power net_W^i and net_H^i.

4.5.1 Finite Horizon Convergence

First, empirical results show that the neuro-dynamic programming solution converges as the number of state space samples used in each stage is increased. In this test reoptimization is performed on the finite horizon case. The first stage begins at hour 07 of August 1, 2013, and the dynamic programming formulation in Figure 4.9 (a) is used to calculate the control policies over the next 72 hours.

Figure 4.10 presents the results. The x-axis is the number of state space samples used in each of the four stages. The y-axis is the MEP of three trials, where the error bars are the standard deviation of the estimate of the MEP. The solid line shows the mean MEP over the three MEP values. As the number of state space samples increases the MEP increases and appears to converge. This indicates that the neuro-dynamic programming solution is converging. Also, the variation among the three trials is large when there are fewer than about 750 samples, but that the variation decreases when there are over 750.

Based upon these results, 1000 state space samples are used in all of the following experiments. This finite horizon test shows that the NDP solution does not

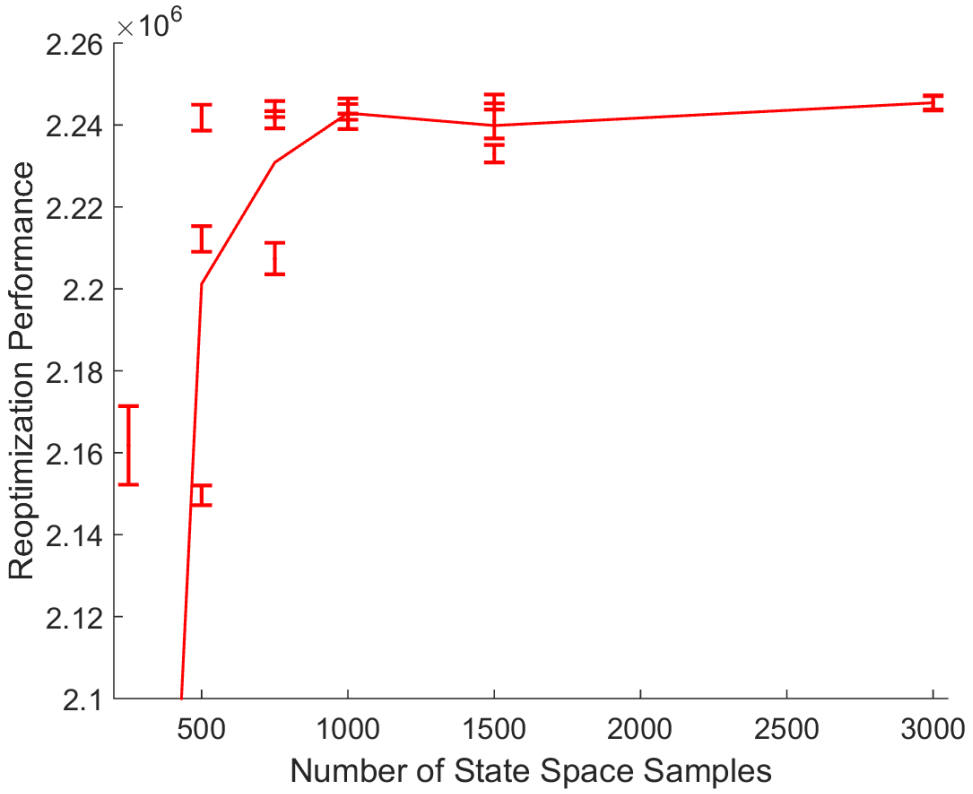


Figure 4.10: Mean Expected Profit (MEP), in Equation (4.31), of finite horizon ADP policies as a function of number of state space samples. The tested number of state space samples are $250, 500, 750, 1000, 1500$ and 3000, and for each number three trials were performed. The MEP are plotted with errorbars, and the solid line shows the mean MEP value over three trials.

become much more accurate when using more samples, and using more samples would linearly increase the computation time required to calculate the solution.

4.5.2 Comparison of Combined and Individual Models

The Combined and Individual models are simulated over a total 8 days, and two trials for each model are performed. The reoptimization procedure uses $1,000$ random initial states for each of the four trials. The reoptimization begins at August 1, 2013 at hour 07, but to avoid any biasing by the initial conditions the

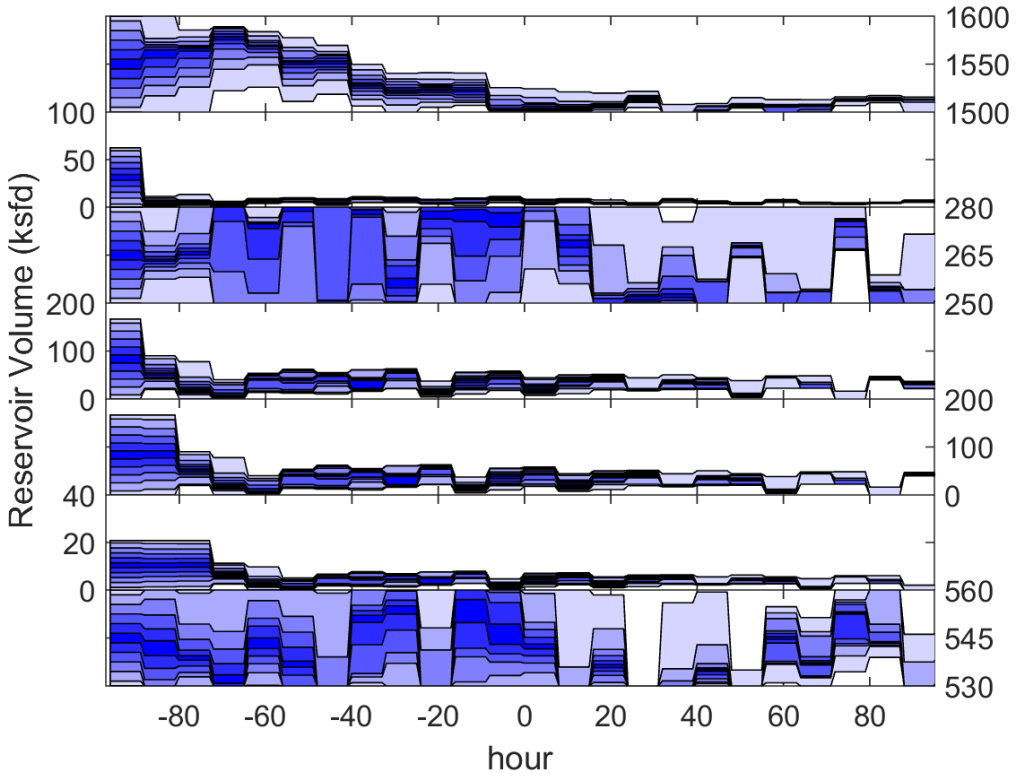


Figure 4.11: Reservoir volumes as a function of time from trial 1 of the Combined model. Time 0 on the x-axis corresponds to August 5, 2013 at hour 07. The i^th subplot shows the volume of reservoir i. The quantile plots show the distribution of reservoir volumes over the $1,000$ reoptimization trajectories.

first four days hours are not included. Therefore, the results are calculated over 4 days beginning on August 5, 2013 at hour 07.

It was decided to exclude the first four days by considering the reservoir volume states shown in Figure 4.11. The initial reservoir volumes are chosen uniformly from within the bounds specified in Table 4.2. The trajectories show that, in particular, the volume of water in reservoir 1 requires about 4 days to settle into a more stable operating regime.

A comparison of the operations in the Combined and Individual models are shown in Figures 4.12 through 4.14. Figure 4.12 shows the hydropower production.

The top and middle quantile plots show the distribution of hydropower production in trial 1 of the Combined and Individual model, respectively. The bottom plot shows the mean value, where the solid and dashed lines are the means of the two trials of the Combined and Individual models.

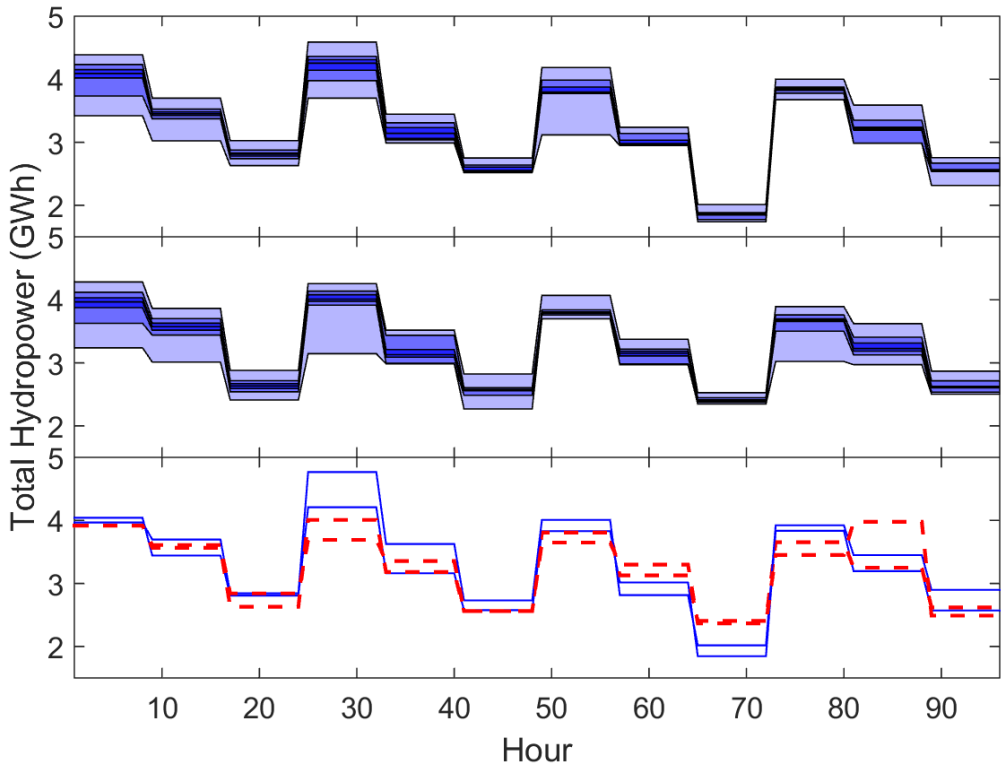


Figure 4.12: Hydropower production. The top and middle plots show the distribution of hydropower production in trial 1 of the Combined and Individual models, respectively. In the bottom plot the solid and dashed lines show the mean hydropower production of the two trials of the Combined and Individual models, respectively.

Figures 4.13 and 4.14 show the power sold on the day and hour ahead markets. The top and middle quantile plots show the distribution of the power sold in trial 1 of the Combined and Individual models, respectively. The bottom plots show the mean values of the two trials, again with solid lines from the Combined model and dashed lines from the Individual model.

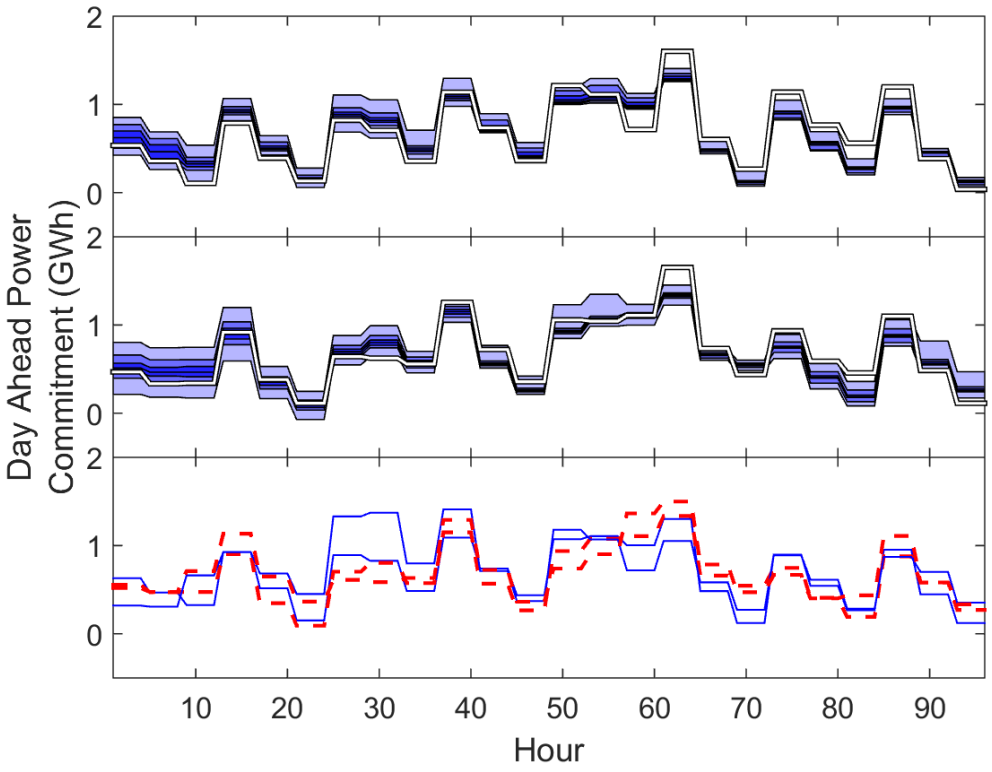


Figure 4.13: Power sold on the day ahead market. Top and middle plots are distribution of da^i in trial 1 of the Combined model and $da^i_W + da^i_H$ in trial 1 of the Individual model, respectively. Black and white lines show results of the simplified model. Bottom plot shows mean value of each trial, with solid lines showing the Combined model.

The thick black and white lines in the top two plots of Figures 4.13 and 4.14 show the day and mean hour ahead power as determined using the simplified model. The maximum amount of hydropower E that can be generated in this simplified model, in Equation 4.30, was selected as $E = 24 \cdot 3.3$ and $E = 24 \cdot 3.25$ GWh in the Combined and Individual models, respectively; these values are selected from Table 4.5, which list the mean hourly hydropower production of the two models. The mean hourly squared error between the power sold in the simplified model and the mean power sold by the dynamic programming solutions are shown in Table 4.4. These results show that the simplified model yields an accurate estimate of

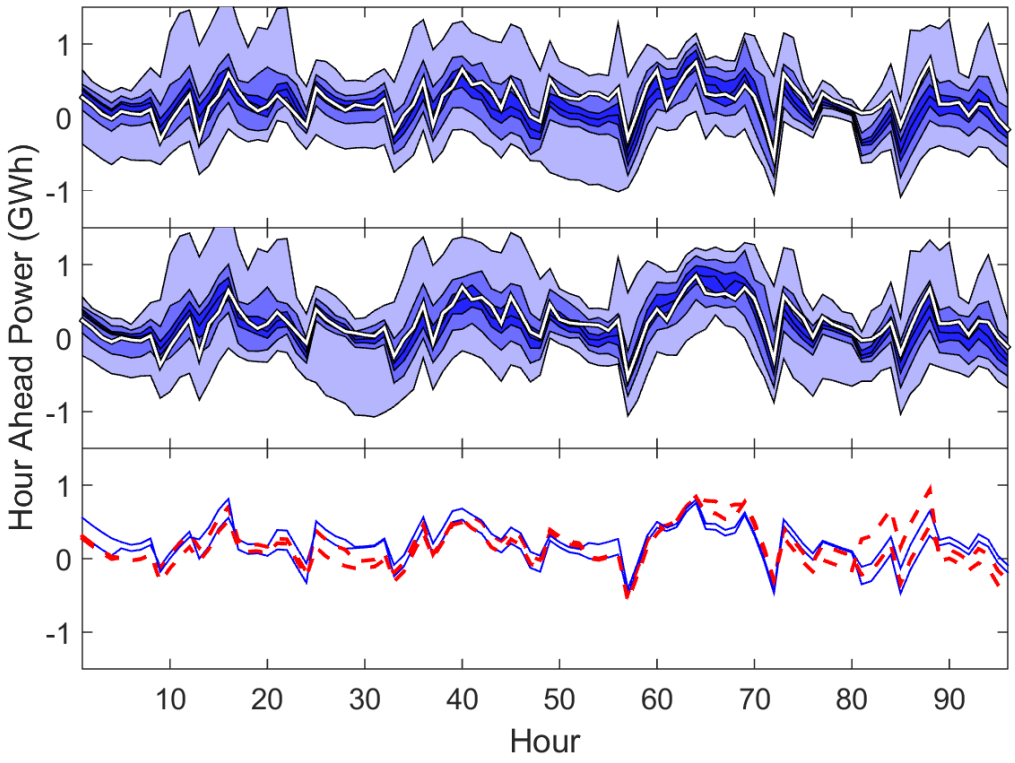


Figure 4.14: Power sold on the hour ahead market. See caption for Figure 4.8.

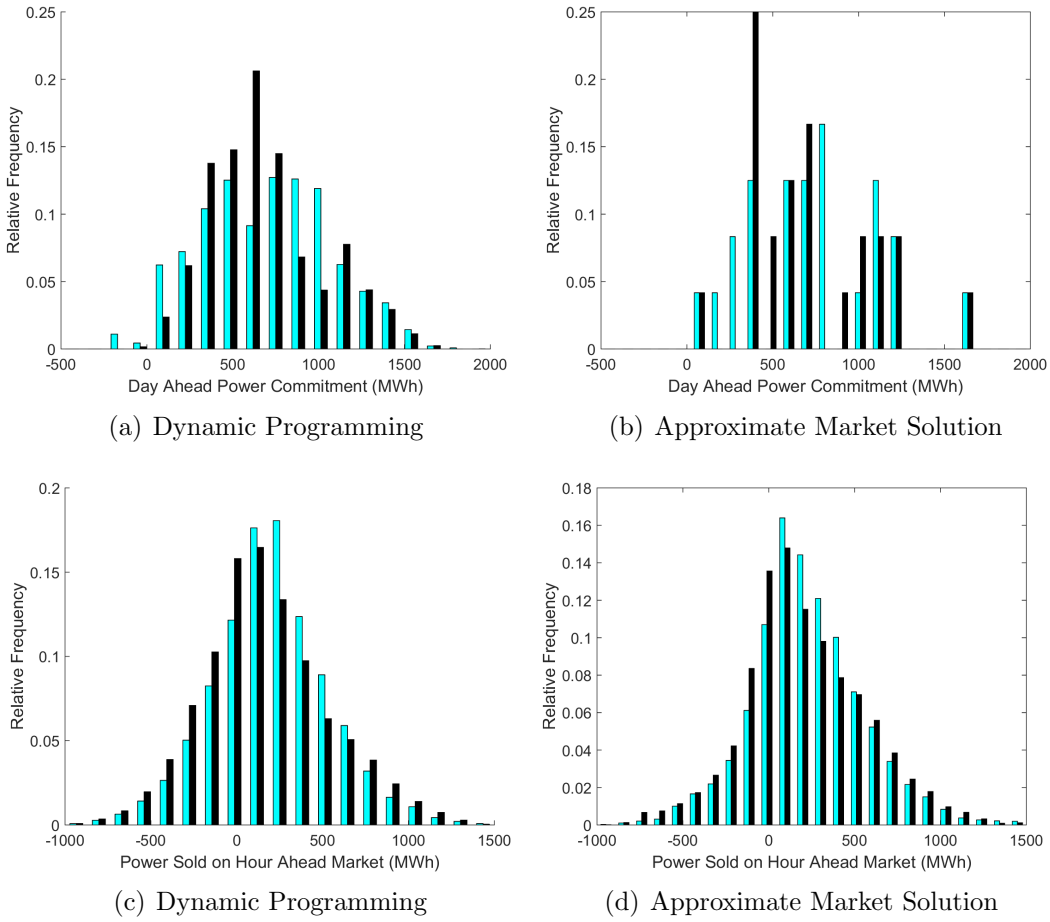
Table 4.4: Square root of mean hourly squared error (MWh) between power sold in the full dynamic programming and simplified models.

	Day Ahead Market	Hour Ahead Market
Combined Model	219	135
Individual Model	134	125

the power sold. Conversely, WH-GENCO and H-GENCO in the Combined and Individual models are able to control the hydropower generation so that the mean values are close to optimal.

These three figures show that the behavior of the single power producer in the Combined model is similar to the combined behavior of the separate W-GENCO and H-GENCO in the Individual model. The net result is that the total power sold on the day and hour ahead markets in the Combined model follows approximately the same trend and distribution as the sum of power sold by W-GENCO and

H-GENCO in the Individual model.



(a) Dynamic Programming

(b) Approximate Market Solution

(c) Dynamic Programming

(d) Approximate Market Solution

Figure 4.15: Distribution of power sold on the day ahead (top) and hour ahead (bottom) markets. Lighter colored bars are from the Combined model, black bars are from the Individual model. Plots on the left are from the dynamic programming model, plots on right use the approximate market solution method.

4.5.3 Comparison of Varying Market Influence

The differences between the two models can be more easily seen by looking at the long-term distribution of power. Figure 4.15 shows the distribution of amount of power sold on the day and hour ahead markets in the Combined and Individual models. These histograms show the relative frequency over all reoptimization

trajectories and over all hours of the four day period in both trials. The plots on the left show the distribution from the full dynamic programming solution, and the plots on right are from the simplified model. The simplified model results are from a simulation over the same four day period. This comparison shows that simplified model is also able to reproduce the same distribution of power sold.

The distribution of power sold on the hour ahead market is narrower in the Combined model, shown as the lighter colored bars, than in the Individual model. This may help to make the power grid more robust by reducing uncertainty of the long-term power generation. The mean power sold in the Combined model is approximately 40 MWh greater every hour than in the Individual model. The distributions of power sold on the day ahead market are similar, although again the Combined model sells approximately 15 MWh more power every hour. This difference in mean hourly production is due to the hydropower generation. It is expected that over a much longer-term simulation the mean hourly power sold in the Combined and Individual models would be approximately the same in both markets because the same amount of water will pass through the reservoirs. Differences would arise if water is spilled through the spillway or if the water heads are significantly different.

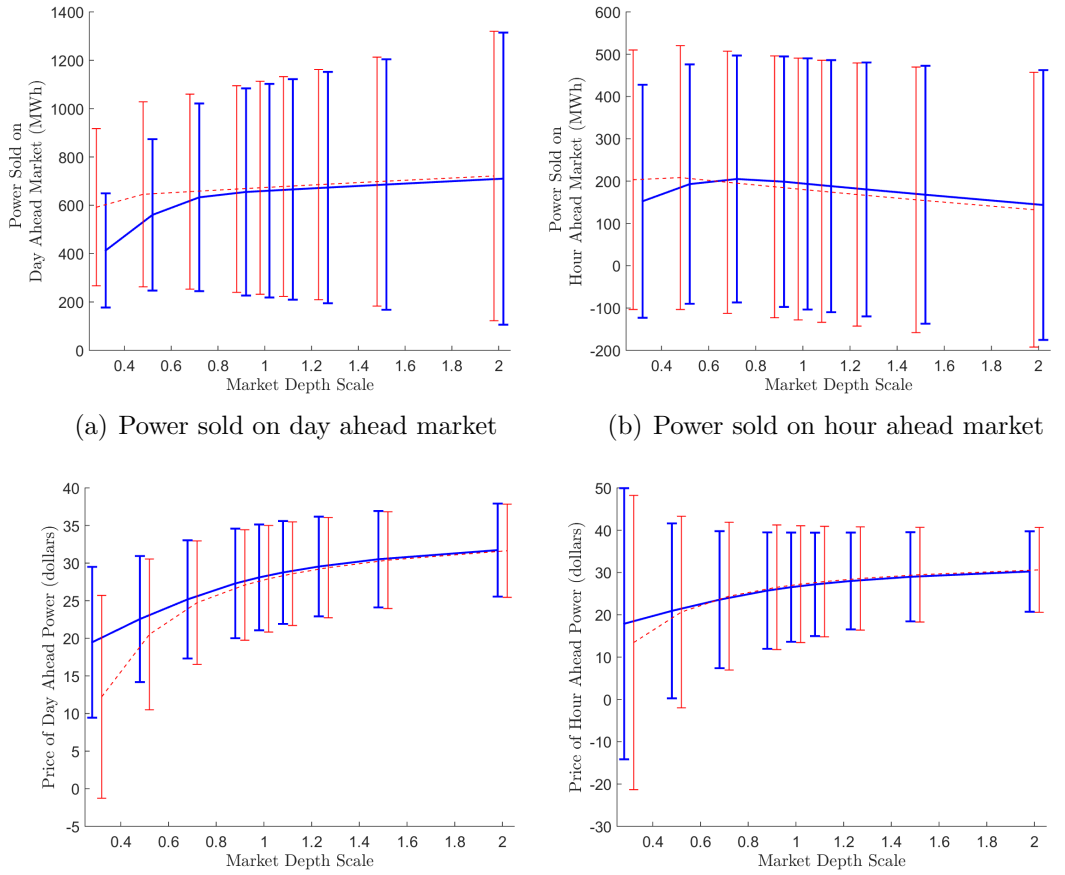
Figure 4.16 shows how these long-term distributions of power production and price of power change as a function of market influence. The thick blue lines are the Combined model results, and the thin dashed red lines are from the Individual model. The data for these plots were generated using the simplified model over a period of 42 days starting on August 1, 2013. The value $E = 24 \cdot 3.275$ MWh was fixed for all days and for both the Combined and Individual model. The errorbars show the standard deviation.

Table 4.5: Summary statistics of the Combined and Individual Models

	Combined WH-GENCO	Individual Sum	Individual H-GENCO	Individual W-GENCO
total profit, 10^6 dollars	1.85	1.66	0.44	1.22
total DA profit	1.74	1.73	0.55	1.19
total HA profit	0.11	-0.07	-0.11	0.03
mean DA price, dollars	26.89	27.03	-	-
mean HA price, dollars	27.03	29.31	-	-
mean power sold, MWh	906.1	851.5	205.1	646.4
mean DA sold, MWh	703.1	687.7	199.0	488.7
mean HA sold, MWh	203.0	163.8	6.0	157.8
mean hydropower production, GWh	3.30	-	3.25	-

The x-axis in these four plots is the market depth scale, which is a multiplicative factor of $depth^i_{DA}$ and $depth^i_{HA}$. A smaller scale gives the power producers more market influence. As the market influence increases the standard deviation of the power sold in the day and hour ahead markets decreases in the Combined model, and is always smaller than in the Individual model. The standard deviation decreases because a profit is only made when the power sold on a market is between 0 and *depth* MWh, so as *depth* decreases WH-GENCO tries to maintain the power sold within a narrower range. In the Individual model H-GENCO does not have the same financial incentive to maintain power production in such a narrow range. Because price is a linear function, the optimal amount of power to sell at any hour i is $depth^i/2$ MWh (where $depth^i$ is the market depth at hour i in either the day ahead or hour ahead market). If W-GENCO has already sold $depth^i/2$ MWh, then H-GENCO still has an incentive to sell power at hour i even though the total profits accrued by W-GENCO and H-GENCO at hour i will only decrease.

Additionally, as the market influence increases WH-GENCO begins to act more like a monopoly. The power production curves show that WH-GENCO decreases hydropower production, thereby reducing total power sold on the day and hour



(a) Power sold on day ahead market

(b) Power sold on hour ahead market

(c) Price of power on day ahead market

(d) Price of power on hour ahead market

Figure 4.16: Distributions of power sold and price of power as a function of market depth scale. A smaller scale means the power producers have more market influence. Thick solid lines and thin dashed lines are the Combined and Individual model, respectively.

ahead markets, in order to keep the prices higher. WH-GENCO is then able to maintain the higher prices, as compared to the Individual model where W-GENCO and H-GENCO do not have the same incentive. Figure 4.17 shows that WH-GENCO is able to make more profit than the sum of W-GENCO and H-GENCO.

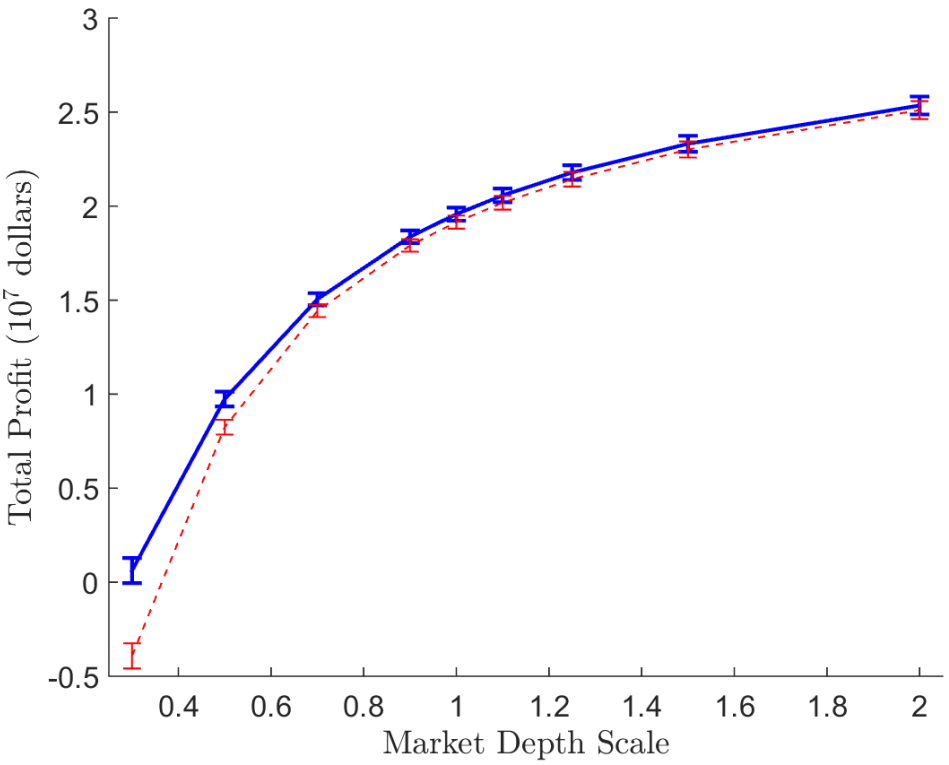


Figure 4.17: Total profit accrued in the Combined model (thick solid line) and in the Individual model (thin dashed line) as a function of market depth. Error bars show standard deviation.

4.6 Conclusions

In this Chapter we developed a model describing how a combined wind and hydropower producer WH-GENCO, an individual hydropower producer H-GENCO, and individual wind power producer W-GENCO participate in the day and hour ahead markets when they have market influence. A dynamic programming formulation was used to determine the optimal amount of power to sell on the day ahead markets and to determine the hydropower reservoir operations. This is a nonconvex and high-dimensional problem, and it was solved using neuro-dynamic programming. Empirical results showed that the amount of power sold by WH-GENCO had a narrower distribution than the sum of power sold by W-GENCO

and H-GENCO, and that these results become more pronounced as the market influence increases. This may help create a more robust power grid. However, WH-GENCO acted more like a monopoly with the increased market influence by selling less power in order to keep power prices higher.

4.A Terminal Value Function

In practice the terminal value function used for a short-term planning model, which has a planning horizon of a few days, is typically calculated by performing a medium- and long-term planning scheduling with a horizon of multiple years [9]. The long-term planning often either consists of solving a discounted infinite horizon problem or uses the results of an infinite horizon problem as the terminal value function. In hydropower problems this long-term value of water is sometimes called the water value, and was first developed in [20]. As an interesting note, [20] essentially performs value function derivative iteration, although it is not formulated using dynamic programming and no proof of convergence is supplied.

In this work the terminal value function used in Equation (4.29) is constructed by calculating the daily mean steady state value of the water reservoir volumes according to the new method developed here. It is similar to an infinite horizon problem. However, the steady state assumption eliminates the need for performing value function iteration.

The main idea is to calculate the total expected daily profit given the constraints that 1) the same day ahead power commitments and reservoir releases must be enacted every day, and 2) the reservoir volumes are periodic, meaning $vol^i = vol^{k24+i}$ for any k and i. The value of the reservoir volumes is then equal to the expected daily value if these decisions are enacted every day.

The first of two variations for calculating the terminal value function of reservoir water volumes vol^7 at hour 07 of any day is

$$V^T(vol^7) = \max_{\substack{da^i \\ i=24,\ldots,31,8,\ldots,23}} \max_{\substack{ph_r^i, sw_r^i \\ r=1,3,7 \\ i=8,\ldots,31}} \left\{ \mathop{\mathbb{E}}_{\substack{wind^i,\, price_{DA}^i,\, price_{HA}^i,\, load^i \\ i=8,\ldots,31}} \right. \\ \left. \Big[\sum_{i=8}^{31} profit_{DA}^i(da^i) + \sum_{i=0}^{23} profit_{HA}^i(net^i) \Big] : vol^7 = vol^{7+24} \right\}. \tag{4.32}$$

There are three main details about this equation. First, notice that the expected value is taken over the wind, day ahead price, Hour Ahead price, and load. This expectation can be calculated by using historical data. We will use a total of thirty day's worth of historical data, so that the expected value is over thirty possible outcomes.

Second, the expected value is not taken over the inflow values because of the steady state constraint. The only way for the reservoir volumes to be periodic while enacting the same daily releases is if the inflow is also periodic. Therefore, the inflow is the mean historical inflow over the same thirty day historical period.

Third, notice the hour indexes. The day ahead power commitment is over hours $24, ..., 31$ and then hours $8, ..., 23$ in order to show that the day ahead power commitments are daily periodic. Normally, the day ahead power is committed over hours $24, ..., 47$, but here the indexes are used to show that the power committed at hour 08 of the next day, i.e. hour 32, is equivalent to the day ahead power committed at hour 08 of the current day.

Equation (4.32) could be used to construct the terminal value function, but there is one main drawback with the current formulation. For every state vol^7 there does not necessarily exist a set of reservoir releases such that $vol^7 = vol^{24+7}$. This can be clearly seen with the following counterexample. In order for the system to be in steady state, the total daily release from each of the three hydropower

reservoirs must each equal the the total daily upstream inflow into reservoir 1. Equivalently, the downstream run-of-river reservoirs cannot have a sum equal to more than the total daily inflow. For example, consider a reservoir volume state $\hat{vol}^7$. If run-of-river reservoirs $4, 5$, and 6 have a combined volume of more than the daily inflow, then this implies the daily release from reservoir 3 is more than the daily inflow, and so there is no control for which $\hat{vol}^{24+7} = \hat{vol}^7$. This constraint means that the value function cannot be calculated for all reservoir volumes in the domain given by Tables 4.2 and 4.3.

Instead, the second variation is developed that can calculate the value function at any state. This variation is

$$V^T(vol^7) = \max_{\substack{da^i \\ i=7+23,\ldots,7+47}} \max_{\substack{ph^i_r, sw^i_r \\ r=1,3,7 \\ i=8,\ldots,7+47}} \left\{ \mathbb{E}^{\substack{wind^i,\, price^i_{DA},\, price^i_{HA},\, load^i \\ i=7+23,\ldots,7+47}} \right.$$
$$\left. \Big[\sum_{i=7+23}^{7+47} profit^i_{DA}(da^i) + \sum_{i=7+23}^{7+47} profit^i_{HA}(net^i) \Big] : vol^{7+23} = vol^{7+47} \right\}. \tag{4.33}$$

In this modification there is a 24 hour period during which the reservoir volume state is allowed to transition to a state for which the steady state value exists, i.e. some volume vol^{7+23} for which there exists a set of releases such that $vol^{7+23} = x^{7+47}$. Notice that the optimization is now over a two day period. In the first 24 hours, from hour 8 to $7 + 23$, there is no profit accrued. Instead, the profit is only accrued over the second 24 hour period, from hour 7+24 to 7+47. The constraint is now that the reservoir volume at the end of the first day, x^{7+23}, is equivalent to the reservoir volume at the end of the second day, x^{7+47}. This formulation allows the reservoir system to spend the first day reaching the optimal steady state volumes.

The addition of the first day prior to the second steady state day ensures that

the reservoir system is able to reach at least one steady state volume. However, it may be the case that in order to reach a steady state volume the hydropower releases may need to violate the release constraints in Table 4.1. Specifically, the three constraints that may need to be violated are rate of change in total release, minimum total release (but the release is still always non-negative), and the maximum total release.

If these constraints are allowed to be violated during the first day then the reservoir system can always reach a steady state volume. At each time step, which is an 8 hour period, let the total release from each of the the 3 hydropower reservoirs be exactly equal to its total inflow from upstream, and call this the 'steady policy'. According to this steady policy the hydropower reservoir volumes do not change. The volume of the run-of-river reservoirs is exactly equal to the the inflow entering the system through reservoir 1 with a time delay. For instance, it takes water 1.5 hours to first reach the segment of river corresponding to run-of-river reservoir 4. In time steps of 8 hours, the volume of reservoir 4 is equal to 6.5/8 of the current time step's inflow into reservoir 1 plus 1.5/8 of the previous time step's inflow. Because it takes water less than 24 hours to traverse the entire system from reservoir 1 to 7, after the first 24 hour period following this steady release policy the reservoir volumes will continue to be daily periodic. Therefore, as long as the daily inflow into reservoir 1 yields a steady policy that satisfies the constraints in Table 4.1 the system has reached a steady state within the first day. Of course, it was verified that the 30 day historical data did result in a daily inflow pattern that yields a steady policy satisfying the constraints.

It may be the case that during the initial 24 hour period the steady policy violates the maximum rate of change in total release constraints or the minimum

total release constraints in Table 4.1. For example, if run-of-river reservoirs 5 and 6 are initialized with volumes $vol_5^7 = vol_6^7 = 0$ then the steady policy would dictate that hydropower reservoir 7 would have a release of exactly 0 during the first 8 hour time step. This would violate the minimum total release constraint. In our reservoir network model the steady policy will never violate the maximum total release constraints because all upstream run-of-river reservoirs have a maximum volume, and therefore maximum release, smaller than the maximum release of the downstream hydropower reservoirs. However, this may not be true in other reservoir networks, and so the maximum total release constraint may need to be violated in other systems.

This difficulty is handled by, if needed, sequentially removing these constraints until it is possible to reach a steady state, meaning there exists a set of releases such that $V^T(s^7)$ can be calculated. Note that these constraints are only removed during the first transition day, and not during the second steady state day. The constraint on change in total release is removed first, followed by the constraint on minimum total release. If the reservoir system was such that the maximum total release constraint also had to be violated, then this third constraint could be removed last.

A total of 4000 data points are sampled from the reservoir volume domain in Tables 4.2 and 4.3 using a quasi random Sobol sequence, and the mean steady state value given by Equation 4.33 is calculated for each. Of these points, the constraint on change in total release was violated in 114, and the constraints on change in total release and minimum total release were violated 16 times. The raw data points are plotted in Figures 4.18 and 4.19 by projecting the data onto a single dimension.

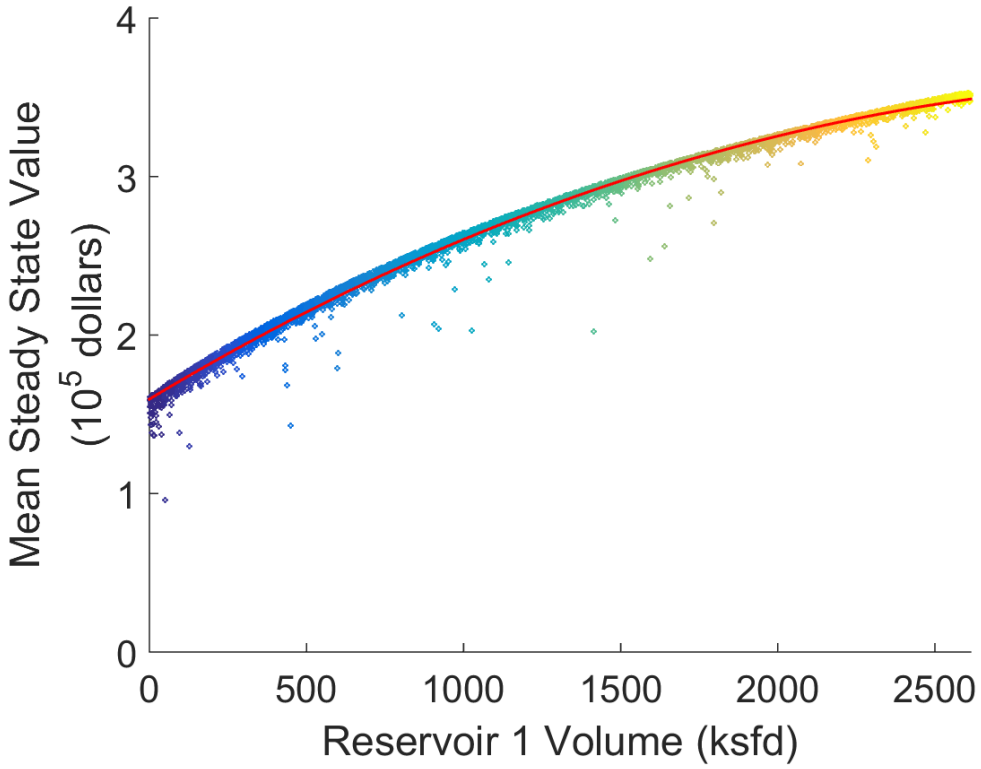


Figure 4.18: Mean steady state value as a function of reservoir 1 volume, vol_1^7. The points are colored according to the x-axis in order to compare to Figure 4.19. The red line shows a slice of the best-fit function in Equation (4.34).

In Figure 4.18 it can be clearly seen that the mean steady state value is strongly a function of the volume of water in reservoir 1, Grand Coulee. Figure 4.19 shows that the value is not as dependent upon the volume of water in the remaining six reservoirs. The data point colors show that most of the variation is explained by vol_1^7.

It was found that the function that can best fit this data is

$$\begin{aligned} V^T([vol_1^7, ..., vol_7^7]) =& (3.6 \cdot 10^6) \log(-5.00 \cdot 10^{-9}(vol_1^7)^2 + 3.46 \cdot 10^{-5} vol_2^7 + 1.05) + \\ & 46.8 vol_2^7 + 51.4 vol_3^7 + 19.6 vol_4^7 + 3.2 vol_5^7 + 0.9 vol_6^7 + 4.5 vol_7^7. \end{aligned} \tag{4.34}$$

The coefficient of determination is $R^2 = 0.981$. A slice of this function is shown in

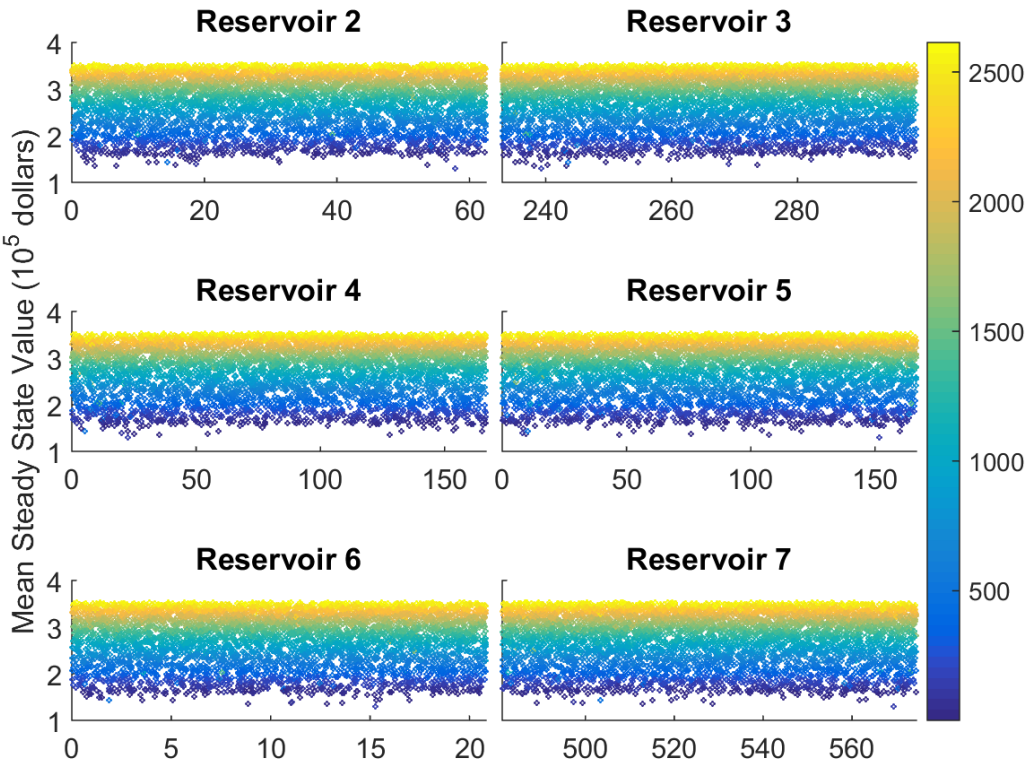


Figure 4.19: Mean steady state value as a function of water volumes in reservoirs 2 through 7. The x-axis is reservoir volume and the y-axis is the mean steady state value. The colorbar shows the volume in reservoir 1, vol_1^7.

red in Figure 4.18. Equation (4.34) is used as the terminal value function in the dynamic programming equations.

BIBLIOGRAPHY

[1] Wind Generation Capacity in the BPA Balancing Authority Area. `https://transmission.bpa.gov/business/operations/wind/WIND\_InstalledCapacity\_PLOT.pdf`. Accessed: 2016-10-20.

[2] Lisias VL Abreu, Mohammad E Khodayar, Mohammad Shahidehpour, and Lei Wu. Risk-constrained coordination of cascaded hydro units with variable wind power generation. *IEEE Transactions on Sustainable Energy*, 3(3):359–368, 2012.

[3] Jorge Márquez Angarita and Julio Garcia Usaola. Combining hydro-generation and wind energy: Biddings and operation on electricity spot markets. *Electric Power Systems Research*, 77(5):393–400, 2007.

[4] Paul D Brown, JA Peças Lopes, and Manuel A Matos. Optimization of pumped storage capacity in an isolated power system with large renewable penetration. *IEEE Transactions on Power systems*, 23(2):523–531, 2008.

[5] Edgardo D Castronuovo and JA Peças Lopes. On the optimization of the daily operation of a wind-hydro power plant. *IEEE Transactions on Power Systems*, 19(3):1599–1606, 2004.

[6] Cristiano Cervellera and Marco Muselli. Efficient sampling in approximate dynamic programming algorithms. *Computational Optimization and Applications*, 38(3):417–443, 2007.

[7] Huajie Ding, Zechun Hu, and Yonghua Song. Stochastic optimization of the daily operation of wind farm and pumped-hydro-storage plant. *Renewable Energy*, 48:571–578, 2012.

[8] Alisha Fernandez, Seth Blumsack, and Patrick Reed. Evaluating wind-following and ecosystem services for hydroelectric dams in pjm. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1897–1906. IEEE, 2012.

[9] Anders Gjelsvik, Birger Mo, and Arne Haugstad. Long-and medium-term operations planning and stochastic modelling in hydro-dominated power systems based on stochastic dual dynamic programming. In *Handbook of power systems I*, pages 33–55. Springer, 2010.

[10] Vincent Guigues and Claudia Sagastizábal. The value of rolling-horizon policies for risk-averse hydro-thermal planning. *European Journal of Operational Research*, 217(1):129–140, 2012.

[11] Ruiwei Jiang, Jianhui Wang, and Yongpei Guan. Robust unit commitment with wind power and pumped storage hydro. *IEEE Transactions on Power Systems*, 27(2):800–810, 2012.

[12] Magnus Korpaas, Arne T Holen, and Ragne Hildrum. Operation and sizing of energy storage for wind power plants in a market system. *International Journal of Electrical Power & Energy Systems*, 25(8):599–606, 2003.

[13] Nils Löhndorf, David Wozabal, and Stefan Minner. Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming. *Operations Research*, 61(4):810–823, 2013.

[14] Xi-Yuan Ma, Yuan-Zhang Sun, and Hua-Liang Fang. Scenario generation of wind power based on statistical uncertainty and variability. *IEEE Transactions on Sustainable Energy*, 4(4):894–904, 2013.

[15] Julija Matevosyan, Magnus Olsson, and Lennart Söder. Hydropower planning coordinated with wind power in areas with congestion problems for trading on the spot and the regulating market. *Electric Power Systems Research*, 79(1):39–48, 2009.

[16] YANG Ming, LIN You, ZHU Simeng, HAN Xueshan, and WANG Hongtao. Multi-dimensional scenario forecast for generation of multiple wind farms. *Journal of Modern Power Systems and Clean Energy*, 3(3):361–370, 2015.

[17] Pierre Pinson, Henrik Madsen, Henrik Aa Nielsen, George Papaefthymiou, and Bernd Klöckl. From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind energy*, 12(1):51–62, 2009.

[18] Didem Sari, Youngrok Lee, Sarah Ryan, and David Woodruff. Statistical metrics for assessing the quality of wind power scenarios for stochastic unit commitment. *Wind Energy*, 2015.

[19] Alexander Shapiro, Wajdi Tekaya, Joari Paulo da Costa, and Murilo Pereira Soares. Risk neutral and risk averse stochastic dual dynamic programming method. *European journal of operational research*, 224(2):375–391, 2013.

[20] Sven Stage and Yngve Larsson. Incremental cost of water power. *Transactions*

of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems, 80(3):361–364, 1961.

[21] Sue Nee Tan. *Computationally Efficient Hydropower Operations Optimization for Large Scale Cascaded Hydropower Systems Reflecting Market Power, Fish Constraints, Multi-Turbine Powerhouses, and Renewable Resource Integration*. PhD dissertation, Cornell University, 2017.

[22] Cheng Wang, Feng Liu, Jianhui Wang, Feng Qiu, Wei Wei, Shengwei Mei, and Shunbo Lei. Robust risk-constrained unit commitment with large-scale wind generation: An adjustable uncertainty set approach. *IEEE Transactions on Power Systems*, 32(1):723–733, 2017.

[23] J Wang, A Botterud, R Bessa, H Keko, L Carvalho, D Issicaba, J Sumaili, and V Miranda. Wind power forecasting uncertainty and unit commitment. *Applied Energy*, 88(11):4014–4023, 2011.

[24] Le Xie and Marija D Ilic. Model predictive economic/environmental dispatch of power systems with intermittent resources. In *Power & Energy Society General Meeting, 2009. PES'09. IEEE*, pages 1–6. IEEE, 2009.

[25] Haoran Zhao, Qiuwei Wu, Shuju Hu, Honghua Xu, and Claus Nygaard Rasmussen. Review of energy storage system for wind power integration support. *Applied Energy*, 137:545–553, 2015.

CHAPTER 5

CONCLUSIONS

In this dissertation we first developed a more efficient approximate dynamic programming algorithm by creating the Fitting via Unimodal Approximation Optimization (FUA) algorithm to more accurately fit the value function with Feedforward Neural Networks with one hidden layer (FFNN1). FUA uses the Unimodal Approximation Optimization (UAO) algorithm we developed to perform a hyperparameter optimization on the number of hidden nodes in FFNN1s. It was empirically demonstrated that the distribution of errors when fitting an FFNN1 is unimodal, but not necessarily convex. Using this observation, UAO was developed to solve this hyperparameter optimization problem that has a discrete domain and a noisy, unimodal objective function. We proved that UAO converges almost surely, and we demonstrated on six test problems that UAO outperforms Bayesian Optimization when applied to fitting FFNN1s. Additionally, FUA only partitions the (state space/value function) data into a training and validation sets, and does not use a test set. It was found that while the FFNN1 could be more accurately estimated using a test set, the fact that a test set was removed from the data resulted in the FFNN1 having a larger error. That is, it is better to have a worse estimate of a smaller FFNN1 error than a more accurate estimate of a larger FFNN1 error.

Altogether, it was demonstrated on three test control problems with 4, 12, and 15 state dimensions that using FUA to fit an FFNN1 to the value function resulted in a more accurate optimal control solution than when using other methods of fitting and FFNN1. This accuracy was evaluated using the results from a reoptimization process, which consists of calculating state space, control, and

cost trajectories over time. The Statistical Accuracy Analysis (SAA) method we created to statistically show that the Mean Expected Cost (MEC) of the optimal control solutions obtained using FUA was smaller (better) than with other methods. Additionally, the Computation and Accuracy Analysis (CAA) demonstrated that FUA created a more optimal solution in less time.

In Chapter 3 we developed the Long Term Generation (LTG) method and the Joint Distribution Comparison (JDC) test to create the long-term synthetic wind scenarios that were used to evaluate control solutions in Chapter 4. LTG generates long-term synthetic wind power scenarios conditioned on sequential short-term historical wind power forecasts. The three main steps of LTG are to create an estimate of the marginal distribution of wind power at each time step conditioned on the forecasts, create the joint distribution, and finally draw scenarios from the joint distribution. Wind scenarios generated by LTG and by the Naive Concatenation Method (NCM) were evaluated using existing methods and the JDC test. JDC tests the null hypothesis that the joint distribution of historical wind power outcomes and historical wind power forecasts is the same as the joint distribution of synthetic wind power outcomes and historical wind power forecasts. This is important because in power systems with wind integration the performance of a control algorithm depends upon the joint distribution of wind power forecast and wind power outcome. Evaluation showed that scenarios generated by the LTG method were more statistically similar to historical data than scenarios generated using NCM.

Finally, in Chapter 4 we developed a stochastic, nonconvex, rolling horizon optimal control model of a wind and power producer with market influence. The data for this power producer is based on the Bonneville Power Administration. The

reservoir system has three hydropower reservoirs, which have the same operating characteristics as Grand Coulee, Chief Joseph, and McNary, and four run-of-river reservoirs to simulate the amount of time it takes water to travel between the hydropower reservoirs. Also, publicly available wind power forecasts and outcomes are used.

The approximate dynamic programming algorithm using FUA was used to solve the control problem, and the policies were evaluated using wind scenarios generated by LTG. Two variations of the power producer were evaluated. In the Combined model a single producer, WH-GENCO, markets both the wind and hydropower, while in the Individual model the wind producer, W-GENCO, and hydropower producer, H-GENCO, act independently. It was found that the total power sold on the day ahead and hour ahead market by WH-GENCO was similar to the sum of W-GENCO and H-GENCO. A simplified model, which only calculates power sold on the markets and not the reservoir operations, was found to be very similar to the controls as calculated by the full dynamic programming formulation. This simplified model was used to show that as the market influence increases WH-GENCO acts more like a monopoly by restricting hydropower production in order to keep power prices artificially high. However, the distribution of power sold by WH-GENCO had a smaller variance, which may help to increase the robustness of the power grid.

There are six immediate ideas for how to extend the work developed in this dissertation. First, the FUA algorithm can be tested on Feedforward Neural Networks (FFNNs) with multiple hidden layers. The first step is to empirically determine whether the mean FFNN error is unimodal with respect to the multiple hyperparameters in FFNNs with multiple hidden layers. If so, then the FUA algorithm can

be applied to fit an FFNN with multiple hidden layers to the value function. Recall that while the UAO algorithm was only tested on one-dimensional problems, the algorithm itself and the proof of convergence hold for multiple dimensions.

The second and third ideas, which are extensions of Chapter 3, are to develop a method for generating a multivariate time series of wind power forecasts and a statistical test to evaluate them. Some work has already been performed towards this goal. It was found that directly generating a multivariate time series of wind power forecasts at each time step did not yield forecast scenarios that were qualitatively similar to the historical data forecasts. The primary difficulty is that the multiple time series are not independent. Instead, a four step procedure seemed promising: a) generate a univariate time series of the wind power outcome; b) generate a univariate time series of the mean forecast error; c) generate multiple independent univariate time series, with series i representing the error of the forecast that is i time steps in advance; d) sum point-wise the time series in step (a) with the time series in (b) with a single time series in (c). The idea behind this approach, which qualitatively appears to work better, is that the multiple time series in step (c) are close to independent.

The last three ideas are all extensions of Chapter 4. First, the model of information used by the independent W-GENCO and H-GENCO can be changed. In particular, as described W-GENCO does not take into account H-GENCO's actions when deciding how much power to sell. For moderate or small market influence this is reasonable, because H-GENCO's actions do not impact the price of power, but this assumption becomes less realistic with a larger market influence. New information models could be investigated, such as where W-GENCO uses a simplified model of H-GENCO in order to determine how H-GENCO's actions will

impact the price of power. Of course, the second step would be to then study the model where H-GENCO, in order to determine its own actions, assumes that W-GENCO uses this updated information model.

Second, the objective function could be altered from an expectation maximization to either a robust optimization or a chance-constrained optimization. Power generating companies may be very sensitive to extreme events. Robust or chance-constrained optimizations, which are more conservative, are often used to formulate the control problem of a power system, and these can be applied to the described wind and hydropower producer.

Finally, the objective function could also be changed from maximizing profit to maximizing some metric of social good. If an electric utility is maximizing profit then that likely means the consumers experience negative effects. For example, if the utility is a price setter, then maximizing profit could mean maximizing price of power, which would mean a higher cost of power for the consumers. Electric utilities are highly regulated, so they usually have competing objectives, such as maintaining a reliable power grid with low cost of power while still maintaining financial stability. There are many ways in which these various objectives, some of which measure social well-being, could be incorporated. For example, the new objective function could be to minimize the daily price of power, thereby saving consumers money, subject to the constraint that the expected weekly profit is non-negative.